Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 172 (2007) 545–587

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Systems Modelling via Resources and Processes: Philosophy, Calculus, Semantics, and Logic

David Pym[1,2]

Hewlett-Packard Laboratories
Bristol, UK

Chris Tofts[3]

Hewlett-Packard Laboratories
Bristol, UK

Abstract

We describe a programme of research in resource semantics, concurrency theory, bunched logic, and stochastic processes, as applied to mathematical systems modelling. Motivated by a desire for structurally and semantically rigorous discrete event modelling tools, applicable to enterprise-scale as well as component-scale systems, we introduce a new approach to compositional reasoning based on a development of SCCS with an explicit model of resource. Our calculus models the co-evolution of resources and processes with synchronization constrained by the availability of resources. We provide a simple denotational semantics as a parametrization of Abramsky's synchronization trees semantics for SCCS. We also provide a logical characterization, analogous to Hennessy-Milner logic's characterization of bisimulation in CCS, of bisimulation between resource processes which is compositional in the concurrent and local structure of systems. We discuss applications to ideas such as location and access control.

Keywords: Resource, semantics, synchrony, process calculus, bunched logic, model checking, discrete event simulation, location, access control, field theory.

Dedicated to Gordon Plotkin on the occasion of his 60th birthday.

1 Introduction: A Modelling Philosophy

This is a paper about a research programme with the objective of delivering mathematical methods for modelling systems that are applicable to large, complex assemblies of resources that are deployed to deliver services. For example, a large corporation may require a uniform global environment with, say, 100 000 desktop

[1] This work was partially supported by a Royal Society Industry Fellowship to Pym and by the University of Bath.

[2] Email: david.pym@hp.com

[3] Email: chris.tofts@hp.com

doi:10.1016/j.entcs.2007.02.020

machines, supported by hundreds of servers, and to connected thousands of other peripheral devices, all connected over a secure global infrastructure of appropriate capacity, capable of accepting remote connections from a diverse range mobile computing platforms. The purpose of such a system is to deliver services to users. For example, the corporation will require support for its business processes, its manufacturing operations, and its research and development work.

At such a scale, we are concerned less with the formal correctness, with respect to a very detailed specification of a small systems component such as microprocessor, say, and more with such questions as performance, reliablity, availability, various aspects of security, and cost. That said, our methods should be able to handle precise correctness arguments where (rarely) necessary.

The intended scale of the modelling task, and the predictive information that we desire to obtain from it, suggest that it is of critical importance to build models at an appropriate level of abstraction [49], both in respect of the structure of the system and of its dynamics.

The dynamics of such systems are, typically, partly determined by their environment. At our intended scale of modelling, however, we cannot possibly hope to capture the precise behaviour of the environment and its effect on the behaviour of the system. A good solution is to use stochastic methods. Specifically, we assume that environmental events occur according to given probability distributions and that components of systems that must process events and data can be described by queueing networks. These assumptions are well-grounded in a great deal of experience in the theory of operating systems and in performance modelling [23,28]. We will develop a brief discussion of these ideas in § 2.

Turning to the structure of systems, the overall view for which we argue is that a system can be modelled effectively as a collection of *locations*, or *places*, at which are found *resources*. Given such a structure, we can execute *processes*, provided they are enabled by the resources available at the required location. The act of executing a process will, in general, cause the structure of the resources, and even their location, to evolve.

So, this is primarily a philosophical paper about a semantically well-founded approach to *practical* systems modelling. Although motivated by enterprise-scale modelling challenges — such as capturing the critical performance and integrity characteristics of wide-area networks — we are concerned to ensure that our techniques should be applicable to addressing component- and protocol-scale correctness questions, as well as systems examples drawn from non-IT domains.

We begin, in § 2, with an a brief account of a discrete event modelling tool, Demos2000 [17], a system which can be seen, in a precise sense, as a partial instantiation of the structural (and, implicitly, stochastic) theory that follows. It also provides an inspiration for our modelling philosophy and mathematical development. In § 3, we introduce a synchronous calculus of resources and processes, **SCRP**, with explicit resources. The process part of the calculus is, *mutatis mutandis*, Milner's [35] synchronous calculus of communicating systems (SCCS). The resource part is based on the Kripke resource monoids that are the basis of the

semantics of the bunched logic **BI** [38,42,45,43,44]. The interaction between the two — that is, the resource access — is handled by functions that describe how actions of the calculus modify the available resources. The operational semantics of **SCRP** exploits the ability of **BI**'s resource semantics to express 'separation condtions', just as in Ishtiaq and O'Hearn's pointer logic [27] and Reynolds' separation logic [46]. We continue, in § 4, to give a range of basic systems examples, intended to demonstrate how to work with **SCRP**. In § 5, we give a simple domain-theoretic denotational semantics for **SCRP**, formulated as a parametrization, on resources, of Abramsky's synchronization trees model [2]. Just as Hennessy-Milner logic [48] provides a logic that is intimately associated with CCS, so there is a modal logic, called **MBI**, that is intimately associated with **SCRP**. In particular, **MBI** provides, via **BI**'s multiplicative conjunction, a logical characterization of concurrent composition which is not available in Hennessy-Milner logic. The basic definition of **MBI** is given in § 6, our systems examples are revisited in § 7, and the essential logical metatheory is described in § 8. In § 10, we explain how our framework can be extended to encompass an explicit, and rather general, notion of location and, in § 11, we explain how our constructions provide a logical analysis of concepts of access control and identity, such as those introduced by Abadi, Burrows, Lampson, and Plotkin [1]. We conclude, in § 12, with a brief speculation on a field theory for resource space.

The basic ideas of **SCRP** and **MBI** — though not the discussion of Demos2000, not the denotational semantics, not the fixed points, not the discussion of location, and not the discussion of security applications — have been presented in [41]. This paper draws directly upon several sections of that paper.

2 An Inspiration: Demos2k

Although our study of the interaction between resources and processes can be motivated from a basic philosophical and mathematical analysis — historically, that is how it happened — it can also be seen as being inspired by the Demos2000 [17] (henceforth Demos2k) system.

Demos2k is a *semantically well-justified* [7,8,9,10,11,17] discrete event simulation language that has been used as a basis for modelling large-scale properties of complex IT systems. A convenient way to understand Demos2k conceptually is as an elegant combination of process algebra and queueing theory. For our purposes, this is best explained by via an example (a classic one, taken from [5]), for which the Demos2k code is given in Figure 1.

Consider a port, with two jetties for docking boats. Each jetty can be used for unloading one boat at a time. Boats arrive at the port periodically and may unload provided a jetty is available and there are two tugs available to take it to the jetty. After unloading, a boat leaves the dock provided a tug is available to remove it from the jetty. The port has three tugs. This informal description is captured for computational evaluation in the model description of Figure 1.

Bearing in mind the formal semantics of Demos2k that has been provided by

```
Cons arrival=negexp(10.0);
Cons docking=2.0;
Cons unloading=normal(14,3);
Cons leaving=2.0;
Cons tug=3;
Cons jetty=2;
Cons simdur=1000;

Res(tugs,tug);
Res(jetties,jetty);

class boat=
{ Entity(Boat,boat,arrival);
   getR(jetties,1);
     getR(tugs,2);
          hold(docking);
     putR(tugs,2);
        hold(unloading);
     getR(tugs,1);
          hold(leaving);
     putR(tugs,1);
   putR(jetties,1);
} (***boat***)

Entity(Boat,boat,0.0);

hold(simdur);
close;
```

Fig. 1. Demos2k code for docking boats [5]

Birtwistle and Tofts [7,8], how can we think of this computational model mathematically, in terms of processes and resources? The model text already provides an immediate clue as to how to consider these elements. The *boats* are represented as **active** entities whose dynamics are presented as *class* definitions. Indeed, in the Demos literature the term entity and process are used interchangeably. In this presentation, the *tugs* and *jetties* are defined as resources which the boats exploit in order to complete their function. The constants within the program simply provide the definitions of duration (specified by the 'hold' command) of certain activities, but do not impact the computational behaviour in any other way.

The recursive call to define a boat entity within the boat class sets up a queue of boats separated by arrival times. The boats then may contend for the limited service resources within the system as a consequence of waiting to be served. The

service discipline in this example is complex, it requires both multiple types of service and multiple instances of the server. Furthermore the queueing entities interact repeatedly with the same serving element, the tugs, but in different ways at different points in their lifecycle.

This choice of active elements is arbitrary. The port model can equally be thought of in terms of tugs that have to combine to provide a boat for a jetty, and jetties which then require a tug to remove the boat. In this view boats are resources which are acted upon by tugs and jetties. Interestingly, this dual model is considerably more difficult to express and to capture within a language like Demos. Whilst a process algebra model appears to avoid this diffculty by regarding all of the elements of the system as active processes, in reality this is not the case. Both abstract versions of the model presented above would have differing accounts within a process algebra. The particular choice of active and passive elements within a model can have major implications for the difficulty of analysis. By making all of the model elements active, the direct process algebra approach does not permit the simple parts of the model to be exploited to simplify calculation. Nor does that approach allow a simple transformation between the multiple presentations which would indicate how simpler calculations could be performed.

Note that, in this model, there is no explicit representation of the structure of the port, that is, of the system. Indeed, in this example, even the implicit representation of the system is very weak, being, quite simply, unnecessary. In other examples, however, that the structure of the system is represented, at least implicitly, is important. For example, in examples concerning systems security, such as when representing aspects of the control of access to various components of a system, the connectivity between the various components must be represented. We return to these ideas in §§ 10 and 11.

We conclude by remarking that our subsequent structural development does not address the question of providing stochastic components comparable to those in Demos2k. Such a development is beyond our present scope. We remark here only that the semantics of Demos2k provided in [7,8,9,10,11] makes clear the sense in which the stochastic components of Demos2k are 'wrapped around' the structural core.

3 A Synchronous Calculus of Resources and Processes

In this section, we describe a simple, synchronous calculus of resources and processes (**SCRP**, for short) based on

(i) the elementary resource semantics that can be interpreted as the basis of the logic of bunched implications (**BI**) [38,42,45,43,44], and

(ii) an appropriate adaptation of Milner's synchronous calculus of communicating systems (SCCS) [35].

Our approach to resource semantics — that is, our account of the notion of resource — has two key aspects. Firstly, it is essentially semantic, not syntactic;

secondly, it is obtained by the classical method of mathematical modelling, that is, we give an axiomatic description of the properties of resources that we require, validating our choice against well-supported examples.

We have found the following axiomatization of resource to be well-justified and useful [38,42,45,43,46,27]:

- A set $\mathbf{R}$ of resource elements;
- A (partial) combination, $\circ : \mathbf{R} \times \mathbf{R} \rightharpoonup \mathbf{R}$ of resource elements;
- A comparison, $\sqsubseteq$, of resource elements; and
- A zero resource element, e.

Mathematically, we can conveniently capture these properties as a preordered partial commutative monoid,

$$\mathcal{R} = (\mathbf{R}\,,\, \circ\,,\, e\,,\, \sqsubseteq),$$

subject to the following bifunctoriality condition relating the monoid structure and the order:

$$\text{for all } r_1,\, r_2,\, s_1,\, s_2, \text{ if } r_1 \sqsubseteq s_1 \text{ and } r_2 \sqsubseteq s_2, \text{ then } r_1 \circ r_2 \sqsubseteq s_1 \circ s_2.$$

Adumbrating some of our technical development, we call such a structure a *Kripke resource monoid*.

This definition is well-supported by a wide range of naturally occurring examples. We discuss a few of them below.

(i) A simple example is provided by the natural numbers, here including 0,

$$\mathcal{N} = (\mathbb{N}\,,\, +\,,\, 0\,,\, \leq),$$

in which combination is given by addition, with unit 0, and comparison is given by less than or equals. In terms of our intended resource semantics, this example may be seen as a basic model of *cost*.

(ii) More general example is given by a monoidal category $(\mathcal{C}, \otimes, I)$, with the order given by the existence of an arrow.

(iii) A computationally richer example is provided by *Petri nets* [45]. Formally, a net

$$\mathcal{N} \;=\; (P\,,\, T\,,\, pre\,,\, post)$$

consists of sets P and T of places and transitions and two functions

$$pre, post : T \to \mathcal{M},$$

from transitions to markings, where a marking is a finite multiset of places and $\mathcal{M}$ denotes the set of all markings. A marking amounts to a function $M : P \to \mathbb{N}$ from places to natural numbers that is zero on all but finitely many places. Addition of markings is given by

$$(M \,+\, N)p \;=\; Mp \,+\, Np.$$

This formalization of Petri nets can be seen as a Kripke resource monoid in several ways. One way is to internalize the reachability relation on markings. If M and N are markings, then define

$$M \Rightarrow N \text{ iff there are } t, M' \text{ s.t. } M = pre(t) + M' \text{ and } N = post(t) + M'.$$

We can then define a preorder, $\sqsubseteq$, on markings by

$$M \sqsubseteq N \text{ iff there are } M_1, \ldots, M_n \text{ s.t. } M = M_1 \Rightarrow \cdots \Rightarrow M_n = N.$$

We let $[-]$, the unit of $+$, denote the empty marking. It follows that $(\mathcal{M}, +, [-], \sqsubseteq)$ is a preordered commutative monoid.

(iv) Finally, the *basic disjointness model*, or BDM, which lies at the core of work on *separation logic* [45,46,27] and which provides a starting point for our subsequent development.

Suppose we are given an infinite set $Res = \{r_0, r_1, \ldots\}$. We think of the elements of Res as primitive resources, or resource IDs, that can be allocated and deallocated. The partial monoid structure is given by taking a world to be a finite subset of Res, and $\circ$ to be union of disjoint sets. In more detail, where $\uparrow$ denotes undefinedness (and, later on, $\downarrow$ definedness),

$$m \circ n = \begin{cases} m \cup n \text{ if } m \cap n = \emptyset \\ \uparrow \qquad \text{otherwise.} \end{cases}$$

The unit of $\circ$ is $\{e\}$, and we take $\sqsubseteq$ to be equality. This example is the basis of Ishtiaq and O'Hearn's pointer logic [27] and Reynolds' separation logic [46].

The composition and ordering structure lifts to sets of resource elements. Let $\wp(\mathbf{R})$ denote the powerset of $\mathbf{R}$ and let $R, S \in \wp(\mathbf{R})$. Then define, for example,

$$R \circ S = \begin{cases} \{\, r \circ s \mid r \in R \text{ and } s \in S \,\} & \text{if each } r \circ s \downarrow \\ \uparrow & \text{otherwise,} \end{cases}$$

with unit $\{e\}$ and, for example,[4]

$$R \sqsubseteq S \text{ iff for all } r \in R \text{ there is } s \in S \text{ such that } r \sqsubseteq s.$$

Equality, $=$, is then given by the symmetric closure of $\sqsubseteq$.

Within a process algebra [35,4,26,36], the common representation of resource is as a separated process. For example, a semaphore is represented as a two-state process, representing whether or not the token is currently available. There have been extensions [12] that attempt to model resource explicitly but these approaches carry both the communication structures of the process algebras alongside the representation of resource. We take the view that resource *is* the fundamental organizing

[4] Note that the ordering on $\wp(\mathbf{R})$ given here is just one of many possible choices; see, for example, [22].

principle of the underlying calculus, an approach taken within process-oriented discrete event languages [5,6]. There has been a demonstration that Milner's calculus SCCS can support a compositional view of resource directly [50]. It is clear, however, that this approach still contains all of the fundamental action structures of SCCS. Our approach is to consider the co-evolution of resources, as discussed above, and processes, in the sense of SCCS, with synchronization being constrained by the availability of resources and with resources being modified by the occurrence of actions.

For our present development, building directly on [41], the sets of resources taken in the BDM are a convenient level of abstraction, for which we shall initially require no further special properties. We might also require that $R \circ S$ be defined only if R and S are disjoint. We write R_1, R_2 for the union of R_1 and R_2, and emphasize that composition is quite different from union. More generally, we might take a more complex structure of resources. For example, we might take $\mathbf{R} = \mathbf{R}_1 \times \ldots \times \mathbf{R}_m$, with a composition $\circ_i$ and ordering $\sqsubseteq_i$ on each $\mathbf{R}_i$.

Our starting point for our calculus of resources and processes is Milner's synchronous calculus of communicating systems, SCCS [35]. Note that the asynchronous calculus CCS is a *sub*-calculus of SCCS [35]. Our main development is to view the statement $E \xrightarrow{a} E'$ as meaning that by using resource required for the action a to be enabled, the process E evolves to E', with a corresponding modification of the available resource. We implement this change of perspective by supposing the existence of a partial *modification* function μ that assigns to each action a and each collection R of resources the collection of resources $\mu(a, R)$ which results from performing a with resource R. Thus our operational rule for action prefix essentially takes the form

$$\frac{}{R\,,\, a : E \xrightarrow{a} \mu(a,R)\,,\, E} \quad \mu(a,R) \text{ is defined.}$$

If the resource R is insufficient for the action a to be enabled, then $\mu(a, R)$ will be undefined.

Synchronization is achieved by requiring that a parallel composition of actions, $a\#b$, be possible only if the resource environment can be decomposed to support a and b separately. Thus our operational rule for parallel composition essentially takes the form

$$\frac{R_1\,,\, E_1 \xrightarrow{a_1} \mu(a_1,R_1)\,,\, E_1' \qquad R_2\,,\, E_2 \xrightarrow{a_2} \mu(a_2,R_2)\,,\, E_2'}{R\,,\, E_1 \times E_2 \xrightarrow{a_1\#a_2} \mu(a_1\#a_2,R)\,,\, E_1' \times E_2'} \quad \begin{array}{l}\mu(a_1\#a_2, R_1 \circ R_2)\\ \text{is defined;}\end{array}$$

that is, it must be possible to decompose R into the resources R_1 and R_2 such that $R = R_1 \circ R_2$, the resources required to support a_1 and a_2 simultaneously, though we admit the possibility of an equality between R_1 and R_2, so allowing sharing as required. Note that synchronization is regulated by resources, in constrast to ACSR [12], in which instantaneous events provide the basic synchronization mechanism. Note also that, in contrast to our local conditions, Gastin and Mislove's

[21] mechanism requires a global construction for synchronization.

An alternative form of presentation of the rule for concurrent composition would, subject to the definedness of the compositions, be the 'functorial' form,

$$\frac{R_1\,,\,E_1 \stackrel{a_1}{\to} R_1'\,,\,E_1' \qquad R_2\,,\,E_2 \stackrel{a_2}{\to} R_2'\,,\,E_2'}{R_1 \circ R_2\,,\,E_1 \times E_2 \stackrel{a_1 \# a_2}{\to} R_1' \circ R_2'\,,\,E_1' \times E_2'},$$

with the restriction on functoriality implied by the coherence condition on the modification function being that

$$R_1' \circ R_2' = \mu(a\#b, R_1 \circ R_2).$$

This convenient presentation is indeed possible, as we shall see at the end of this section.

One fundamental consequence of this approach is that we should wish to maintain all of the interactions that lead to the current resource use transition within a process. In some sense, we need to know how the current resource utilization can be decomposed. Consequently, we must abandon the elegant use of the free abelian group of actions within SCCS to describe actions, restricting ourselves to the more basic free abelian monoid [35],

$$\mathcal{A} = (Act\,,\,\#\,,\,\mathbf{1}).$$

If we were to take an abelian group, then an action a might result from the composition $a\#b^{-1}$ and b thus, in some sense, making use of more resource (as the $b\#b^{-1}$ becomes hidden). Taking resource as the basic organizing principle, this form of hiding makes decomposition difficult to track. Nevertheless, our formulation permits the formulation of compound atomic actions that are able to emulate the difficult wait-until construct of discrete event simulation languages such as Demos [5].

SCCS, in common with CCS, uses a notion of *restriction*. In our setting, a more natural concept is that of a *local* action, in which a collection of resources is available only to the process to which it is bound. Informally, the operational rule should take the form

$$\frac{R \circ S\,,\,E \stackrel{a}{\longrightarrow} R' \circ S'\,,\,E'}{R\,,\,(\nu S)E \stackrel{(\nu S)a}{\to} R'\,,\,(\nu S')E'},$$

where '$(\nu S)a$' denotes the action a without the components of it that are associated with the bound resource S. These components are 'hidden' in the subsequent evolution.

The calculus described here, first formulated in [41], is called the *synchronous calculus of resource processes*, or **SCRP** (pronounced 'scrap'). Following the notation of SCCS and the π-calculus, we present the syntax of the process element of the calculus. The finite part of the language is as follows:

- **1** — the unit action;
- $a : E$ — a process that performs the action a to become the process E;

- $E + F$ — a process that evolves as E or as F, unit $\mathbf{0}$;
- $E \times F$ — a process that synchronously uses resources as E and F;
- $(\nu S)E$ — a process with a local, or hidden, evolution relative to resource S and modification function μ, as explained below.

We suppress relabelling, treating it as a metatheoretic operation.

The recursive part of the language has the following syntax:

- Process variables, X, and fixed points, $\mathrm{fix}X.E$.

For practical purposes, we work with definitions of constants, $C \stackrel{\mathrm{def}}{=} E$, rather than fixed points.

Unlike for standard process algebras, we must, as we have seen, define the environment of resources wherein the process evolves. This is given as a set of permitted actions drawn from the monoid. Thus our operational judgement is essentially of the form

$$R\,,\, E \xrightarrow{a} R'\,,\, E'$$

and is intended to be read as 'process E evolves via action a relative to the set of resources R'. R is assumed to be a set of elements of resource drawn from a Kripke resource monoid. But this alone is not sufficient. As we have seen, we must set up an association between the actions of the monoid and the resources in the system. For now, we take the following:

- a partial function $\mu : Act \times \wp(\mathbf{R}) \rightharpoonup \wp(\mathbf{R})$, with
- $\mu(\mathbf{1}, R) = R$ and, if $\mu(a, R) \downarrow$ and $\mu(b, S) \downarrow$, then $\mu(a\#b, R \circ S) = \mu(a, R) \circ \mu(b, S)$, and,
- for all a and R, the identity $\mathrm{id}(a, R) = R$.

In our definition of **SCRP**'s operational semantics, we omit explicit mention of the partial function μ wherever possible. We write $\mu^{-1}S$ for the set of actions a such that $\mu(a, S) \downarrow$. In order to give the rule for hiding, we need an auxiliary definition of *deletion*:

- Let $A = \{a_1, \ldots, a_m\} \subseteq Act$ be a finite set of actions. Then define

$$\Pi A = \Pi_{i=1}^{m} a_i = a_1 \# \ldots \# a_m.$$

If b is the product over a subset of A, then we refer to 'b in ΠA'.

- Let αs and βs denote atomic actions. Let

$$a = {\alpha_1}^{k_1} \# \ldots \# {\alpha_m}^{k_m}$$
$$b = {\beta_1}^{l_1} \# \ldots \# {\beta_n}^{l_n}.$$

Then define

$$a \,/\, b = \Pi_{i=1}^{m} \{\alpha_i^{k_i} \mid \text{for all } 1 \leq j \leq n,\ \alpha_i \neq \beta_j\}.$$

$$Act \quad \frac{}{R\,,\ a : E \xrightarrow{a} \mu(a,R)\,,\ E} \quad \mu(a,R)\downarrow$$

$$Prod \quad \frac{R\,,\ E \xrightarrow{a} \mu(a,R)\,,\ E' \quad S\,,\ F \xrightarrow{b} \mu(b,S)\,,\ F'}{R\circ S\,,\ E\times F \xrightarrow{a\#b} \mu(a\#b, R\circ S)\,,\ E'\times F'}$$

$$Sum_i \quad \frac{R\,,\ E_i \xrightarrow{a} \mu(a,R)\,,\ E_i'}{R\,,\ E_1+E_2 \xrightarrow{a} \mu(a,R)\,,\ E_i'} \quad i=1,2$$

$$Hide \quad \frac{R\circ S\,,\ E \xrightarrow{a} R'\circ S'\,,\ E'}{R\,,\ (\nu S)E \xrightarrow{a\,/\,\Pi\mu^{-1}S} R'\,,\ (\nu S')E'} \quad \mu(a\,/\,\Pi\mu^{-1}S, R) = R'$$

Table 1
Operational Semantics of Finite **SCRP**

For theoretical and semantical purposes, we take recursive process terms to be defined by a fixed-point rule in the operational semantics.

$$Rec \quad \frac{R\,,\ E[\mathrm{fix}X.E/X] \xrightarrow{a} \mu(a,R)\,,\ E'}{R\,,\ \mathrm{fix}X.E \xrightarrow{a} \mu(a,R)\,,\ E'}$$

Table 2
Operational Semantics of Recursive **SCRP**

In practice, following Milner [36], we define recursive processes using the rule for constants:

$$Con \quad \frac{R\,,\ E \xrightarrow{a} \mu(a,R)\,,\ E'}{R\,,\ C \xrightarrow{a} \mu(a,R)\,,\ E'} \quad C \stackrel{\mathrm{def}}{=} E.$$

The system obtained is, evidently, equivalent to the one obtained using the fixed-point rule.

Definition 3.1 Bisimulation, $\sim_\mu$, is the largest binary relation on resource–process pairs, $R\,,\ E$ such that if $R\,,\ E \sim_\mu R\,,\ F$, then

(i) $R\,,\ E \xrightarrow{a} \mu(a,R)\,,\ E'$ implies, for some F',

$$R\,,\ F \xrightarrow{a} \mu(a,R), F' \text{ and } \mu(a,R)\,,\ E' \sim_\mu \mu(a,R)\,,\ F',$$

and

(ii) $R\,,\ F \xrightarrow{a} \mu(a,R)\,,\ F'$ implies, for some E',

$$R\,,\ E \xrightarrow{a} \mu(a,R)\,,\ E' \text{ and } \mu(a,R)\,,\ E' \sim_\mu \mu(a,R)\,,\ F'.$$

We shall need the familiar property that bisimulation is a congruence, that is, in our setting, that if, for all R, $R\,,\ E \sim_\mu R\,,\ F$, then, for all evident terms R, a, G, and S, $R\,,\ a : E \sim_\mu R\,,\ a : F$, $R\,,\ E+G \sim_\mu R\,,\ F+G$, $R\circ S\,,\ E\times G \sim_\mu R\circ S\,,\ F\times G$, and $R\,,\ (\nu S)E \sim_\mu R\,,\ (\nu S)F$.

Proposition 3.2 *Bisimulation is a congruence.*

Proof. Straightforward arguments by induction on the structure of resource–process pairs establish that $\sim_\mu$ is both an equivalence relation and substitutive. □

We conclude this section with a formal statement of the simple but rather useful property that the resource modifications caused by actions are entirely determined by μ:

Proposition 3.3 *Let a be an action with modifications $\mu(a,-)$. If $R\,,\,E \xrightarrow{a} R'\,,\,E'$, then $R' = \mu(a,R)$.*

Proof. By induction on the structure of derivations in the operational semantics of **SCRP**, the (critical) base case being immediate.

For an example inductive step, suppose the last rule applied is *Prod*,

$$\frac{R\,,\,E \xrightarrow{a} R'\,,\,E' \quad S\,,\,F \xrightarrow{b} S'\,,\,F'}{R \circ S\,,\,E \times F \overset{a\#b}{\to} R' \circ S'\,,\,E' \times F'}.$$

By the induction hypothesis, we have $R' = \mu(a,R)$ and $S' = \mu(b,S)$. So, $R' \circ S' = \mu(a,R) \circ \mu(b,S)$ and so, by the coherence conditions on μ, $R' \circ S' = \mu(a\#b, R \circ S)$.□

In the sequel, where no significant confusion can occur, we write just R' for $\mu(a,R)$.

4 Some Systems Examples

We present some examples, taken from [41], to illustrate the commonly required interactions in a concurrent setting:

- mutual exclusion;
- resource transfer;
- handshaking;
- private channels;
- asynchronous handover.

These examples — see [3] for a discussion of the basic components of concurrent systems — have been chosen to illustrate how we can specify core concurrent systems concepts in this simple setting.

The first four examples can be stated clearly enough without giving a detailed description of the Kripke resource monoids involved. For the last example — asynchronous handover — we take the following Kripke resource monoid: assume basic sets of resource elements R, writing R^n for $n \geq 0$ distinct copies of R, that is, $\underbrace{R \circ \ldots \circ R}_{n\ times}$, with $R^0 = \{e\}$. If we take the ordering on resources to be $R^m \sqsubseteq R^n$ iff $m \leq n$, then bifunctoriality follows immediately and, up to some requirements

about the definedness of the modification function and the interpretation of predicates, which we discuss in § 6, we get an essentially intuitionistic model. If we were to take the discrete ordering, which is not appropriate for all examples, we should obtain an essentially classical model, a situation similar to that which obtains in separation logic [46].

Although we have explained in [41] that taking 'enabling' functions — that specify the minimum resources required for an action to occur — as distinct from the modification functions is logically problematic, we nevertheless find it helpful to use the idea as a notational abbreviaton. Specifically, we write

$$\rho(a) = \min \{R \mid \mu(a, R) \downarrow\}$$

where such a minimum exists. This abbreviation is particularly convenient in the context of the Kripke resource monoid, described above, used in the asynchronous handover example.

Mutual Exclusion

In this example, we have two (or more) processes and some points in the computation at which at most one of them may be active, often termed a critical region or section. To that end we define a process in the following manner:

$$E \stackrel{\text{def}}{=} nc : E + critical : E_{critical}$$
$$E_{critical} \stackrel{\text{def}}{=} critical : E_{critical} + critical : E$$

To give sufficient detail, we suppose that the minimum resources required for the various actions to be enabled to be given by $\rho(nc) = \{e\}$ (nc is 'not critical'), and $\rho(critical) = \{R\}$. Now, the resource process

$$R\,,\, E \times E,$$

where $\mu(a, R) = R$, for all a, defines a system exhibiting mutual exclusion. The important point is that, in the application of the $Prod$ rule, we have $R = R \circ \{e\}$. In other words, the resource is available as itself at the same time as an empty resource. Consequently the evolutions of our system are as follows:

$$R\,,\, E \times E \stackrel{nc\#nc}{\rightarrow} R\,,\, E \times E$$

$$R\,,\, E \times E \stackrel{nc\#critical}{\rightarrow} R\,,\, E \times E_{critical}$$
$$R\,,\, E \times E_{critical} \stackrel{nc\#critical}{\rightarrow} R\,,\, E \times E_{critical}$$
$$R\,,\, E \times E_{critical} \stackrel{nc\#critical}{\rightarrow} R\,,\, E \times E.$$

Notice that at no point is the action $critical\#critical$ performed nor do we see both processes in the state $E_{critical} \times E_{critical}$.

Note that in this example the resource we have used plays the role of a semaphore; this demonstrates the capability of the calculus to exploit resource directly. A more standard process algebra solution to this problem is to use communication via handshaking; we shall demonstrate how that is achieved within this calculus in our next example.

Resource Transfer

As an extension of the semaphores example, above, we may wish to establish a system in which only one of the parallel tasks is 'active' at any one time, but in which the tasks take turns. One way of achieving this is described below.

We take as resources sets R_1 and R_2. Then we define the following modification functions:

$$\mu(put_1, R_1) = R_2 \qquad \mu(get_1, R_1) = R_1$$
$$\mu(put_2, R_2) = R_1 \qquad \mu(get_2, R_2) = R_2.$$

We take the following minimal requirements, $\rho(get_1) = R_1$ and $\rho(get_2) = R_2$. We also take $\rho(put_1) = R_1$ and $\rho(put_2) = R_2$.

We take the following process definitions:

$$E1 \stackrel{\text{def}}{=} get_1 : E1_{critical} + \mathbf{1} : E1$$
$$E1_{critical} \stackrel{\text{def}}{=} \mathbf{1} : E1_{critical} + put_1 : E1$$
$$E2 \stackrel{\text{def}}{=} get_2 : E2_{critical} + \mathbf{1} : E2$$
$$E2_{critical} \stackrel{\text{def}}{=} \mathbf{1} : E2_{critical} + put_2 : E2,$$

where, of course, we take $\rho(\mathbf{1}) = \{e\}$.

Then the system $R_1 \circ \{e\}\,,\ E1 \times E2$ represents one in which the processes $E1$ and $E2$ exchange ownership of a resource in order that they may enter their respective critical sections:

$$R_1\,,\ E1 \times E2 \stackrel{get_1 \# \mathbf{1}}{\rightarrow} R_2\,,\ E1_{critical} \times E2.$$

There are two clear generalizations of this system: we can extend the number of resources whilst keeping the same pass-on property and so obtain a 'round-robin' scheduler [36]; alternatively, we can extend the processes as follows:

$$E1 \stackrel{\text{def}}{=} get_1 : E1_{critical} + swap_1 : E1 + \mathbf{1} : E1,$$

with $\mu(swap_1, R_1) = R_2$ and we have $\rho(swap_1) = R_1$, and the obvious symmetric definitions. We then obtain a mutual exclusion system with a designated, or transferable, token.

It should be clear from these examples that the modification functions, of the form $\mu : Act \times \wp(\mathbf{R}) \rightharpoonup \wp(\mathbf{R})$, allow very flexible models of resource transfer between concurrent processes.

Handshaking

In this example, we desire that two processes should proceed only if they mutually agree on progress. In other words, there is a point in the computation that is preceded by a point of mutual agreement or a 'join'. The following process definitions will illustrate the point:

$$E_1 \stackrel{\text{def}}{=} wait_{E_1} : E_1 + go_{E_1} : E_1'$$
$$E_2 \stackrel{\text{def}}{=} wait_{E_2} : E_2 + go_{E_2} : E_2'$$

we take $\rho(go_{E_1}) = R_1$ and $\rho(go_{E_2}) = R_2$ but *importantly* $R = R_1 \circ R_2$, remembering that this is *not* the same as R_1, R_2 (recall that we use this list notation for the union of sets of resources) and can only be 'split up' (composition is *not*, in general, union) by a use of a *Prod* rule. We take $\rho(wait_1) = \rho(wait_2) = \{e\}$.

The evolutions of $R\,,\, E_1 \times E_2$ are as follows:

$$R\,,\, E_1 \times E_2 \stackrel{wait_{E_1} \# wait_{E_2}}{\longrightarrow} R\,,\, E_1 \times E_2$$

$$R\,,\, E_1 \times E_2 \stackrel{go_{E_1} \# go_{E_2}}{\longrightarrow} R\,,\, E_1' \times E_2'.$$

Notice that E_1 and E_2 either wait or proceed together. Obviously in a larger system the states E_1 and E_2 need not be arrived at at the same time.

Privacy

In the foregoing example, we may wish to ensure that *only* E_1 and E_2 can interact by using the composed resource. So we would form the process

$$(\nu R_1 \circ R_2)(E_1 \times E_2),$$

with E_1, E_2, and ρ taken as above. Note that the requirement that that each go_{E_i} be enabled by the available resource leads, essentially, to the association of each R_i and E_i.

Asynchronous Handover

The classic form of this example is the producer–consumer problem: that is, there is one process that can generate work and leave it for another process to handle later. Consider the following process definitions:

$$Prod \stackrel{\text{def}}{=} nowork : Prod + work : Prod$$
$$Cons \stackrel{\text{def}}{=} wait : Cons + cons : Cons,$$

where we require $\rho(nowork) = \{e\}$, $\rho(wait) = \{e\}$, $\rho(work) = \{e\}$, $\rho(cons) = R$ and, writing R^n for $n > 0$ distinct copies of R, *i.e.*, $\underbrace{R \circ \ldots \circ R}_{n\ times}$,

$$\begin{aligned} \mu(nowork, \{e\}) &= \{e\} & \mu(nowork, R^n)vv &= R^n \\ \mu(wait, \{e\}) &= \{e\} & \mu(wait, R^n) &= R^n \\ \mu(work, \{e\}) &= \{R\} & \mu(work, R^n) &= R^{n+1} \\ & & \mu(cons, R^n) &= R^{n-1}. \end{aligned}$$

It follows that the system

$$\{e\}\,,\ Prod \times Cons$$

behaves as a producer–consumer system with a counter R. Given a generic state of the system, we have the following evolutions

$$\{e\}\,,\ Prod \times Cons \overset{nowork\#wait}{\rightarrow} \{e\}\,,\ Prod \times Cons$$

$$\begin{aligned} \{e\}\,,\ Prod \times Cons &\overset{work\#wait}{\rightarrow} R\,,\ Prod \times Cons \\ R^n\,,\ Prod \times Cons &\overset{nowork\#wait}{\rightarrow} R^n\,,\ Prod \times Cons \\ R^n\,,\ Prod \times Cons &\overset{nowork\#cons}{\rightarrow} R^{n-1}\,,\ Prod \times Cons \\ R^n\,,\ Prod \times Cons &\overset{work\#cons}{\rightarrow} R^n\,,\ Prod \times Cons \\ R^n\,,\ Prod \times Cons &\overset{work\#wait}{\rightarrow} R^{n+1}\,,\ Prod \times Cons. \end{aligned}$$

5 A Simple Denotational Semantics

We provide a denotational semantics for **SCRP** that is fully abstract with respect to the evident adaptation of the usual operational preorder [35,24,2] to **SCRP**. Our semantics is a parametrization on resources of Abramksy's construction in **SFP** [2,39,40] of a domain of synchronization trees for SCCS. Our parametrization amounts to the inclusion of the resource constraints on SCCS imposed by **SCRP**, the key cases being action prefix, concurrent composition, and hiding.

Abramsky's paper provides a fully detailed construction of the domain, $\mathcal{D}$, of synchronization trees and a proof of full abstraction for SCCS. Rather than repeating the details of Abramsky's construction, we briefly sketch the key points together with the modifications that we require.

For the purposes of this section, we take an explicit account of undefined processes, which we have so far elided. We assume the existence of a basic undefined process, Ω (following [2]), with the basic operational assumption

$$\Omega \quad \overline{R\,,\ \Omega\uparrow}.$$

We then take the evident rules expressing undefinedness for each of the combinators. For example,

$$\frac{R\,,\ E_i \uparrow}{R\,,\ E_1 \diamondsuit E_2 \uparrow} \quad i = 1, 2,$$

where $\diamondsuit$ is $+$ or $\times$. Hiding is handled similarly. We also suppose, for the purposes of this section, that each $\mu(a, -) : \wp(\mathbf{R}) \to \wp(\mathbf{R})$ is continuous (*i.e.*, monotonic and preserves lubs of ω-chains, so that $\mu(a, \bigcup_i R_i) = \bigcup_i \mu(a, R_i)$).

We begin with the necessary operational preorder [24,2] for **SCRP**. The symmetric closure of this relation is the (strong) bisimulation taken in Definition 3.1.

Definition 5.1 [operational preorder] Let $R\,,\ E$ and $R\,,\ F$ be closed **SCRP**-terms. We define the operational pre-order $R, E \preceq^{\mu} R, F$ is defined inductively as follows, where α is an ordinal:

(i) $R\,,\ E \preceq^{\mu}_{0} R\,,\ F$, for all R, E, F;

(ii) $R\,,\ E \preceq^{\mu}_{\alpha+1} R\,,\ F$ if, for every, $a \in Act$,
  (a) $R\,,\ E \xrightarrow{a} \mu(a, R)\,,\ E'$ implies, for some F' s.t. $R\,,\ F \xrightarrow{a} \mu(a, R)\,,\ F'$ and $\mu(a, R)\,,\ E' \preceq^{\mu}_{\alpha} \mu(a, R)\,,\ F'$, and
  (b) $E \downarrow$ implies
    $F \downarrow$, and
    $R\,,\ F \xrightarrow{a} \mu(a, R)\,,\ F'$ implies, for some E' s.t. $R\,,\ E \xrightarrow{\mu} (a, R)\,,\ E'$ and $\mu(a, R)\,,\ E' \preceq^{\mu}_{\alpha} \mu(a, R)\,,\ F'$;

(iii) For a limit ordinal λ, $R, E \preceq^{\mu}_{\lambda} R, F$ iff, for all $\alpha < \lambda$, $R, E \preceq^{\mu}_{\alpha} R, F$;

(iv) $R\,,\ E \preceq^{\mu} R\,,\ F$ if, for every $\alpha \geq 0$, $R\,,\ E \preceq^{\mu}_{\alpha} R\,,\ F$.

Processes are interpreted as continuous functions on the powerset monoid of resources, $\wp(\mathbf{R})$. So, relative to an interpretation I of process variables, we have

$$[\![E]\!]^{\mathcal{D},I}_{\mu} : \wp(\mathbf{R}) \to \mathcal{D},$$

where $\mathcal{D}$ is Abramsky's domain of synchronization trees for SCCS [2]. We drop the superscript I when considering just finite terms.

The key differences between our semantics for **SCRP** and Abramsky's semantics for SCCS, arising from the evident parametrization on resources, are the following:

- **SCRP**'s resource structuring ensures that the formation of concurrent composition is regulated by the composition of the resource structures of the components of the composition;
- **SCRP**'s use of resource hiding, or local resources, eliminates the need for restriction, the relationship with actions being maintained via **SCRP**'s enabling functions.

Neither of these variations, however, requires a modification of Abramsky's domain construction. Rather, they are handled by the parametrization on $\wp(\mathbf{R})$. Accordingly, we sketch Abramsky's construction and full abstraction proof for SCCS, explaining our required modifications as we proceed.

Abramsky's construction proceeds in the category **SFP** [39]. **SFP** is a category of algebraic domains that is closed under the constructions listed below.

- *Separated sums.* $\sum_{a\in A} D_a$ is formed by taking the disjoint union of a countable family of domains. Elements are written as $\langle a, d\rangle$, where $a \in A$ and $d \in D_a$. This construction is the key to the interpretation of action prefix in the synchronous setting.
- *The Plotkin powerdomain.* $P[D]$ — for domain D with order $\sqsubseteq$ — with the Egli-Milner order:
  for $X, Y \in D$, $X \sqsubseteq_{EM} Y$ iff, for all $x \in X$, there exists $y \in Y$ s.t. $x \sqsubseteq y$ and, for all $y \in Y$ there exists $x \in X$ s.t. $x \sqsubseteq y$.

  We need to recall some auxiliary definitions from domain theory [40]. Let $X \subseteq D$.

(i) $Con(X) \stackrel{\text{def}}{=} \{\ d \mid \text{there are } d_1, d_2 \in X \text{ s.t. } d_1 \sqsubseteq d \sqsubseteq d_2\ \}$.

(ii) $X^* \stackrel{\text{def}}{=} Con \circ Cl(X)$, where $Cl(X)$ is the closure associated with the Lawson topology [2].

X is *convex closed* if $X = Cl(X)$ and is *closed* if $X = X^*$.

$P[D]$ has associated with it several continuous operations:

· P is *functorial*: if $f : D \to E$, then $P(f) : P[D] \to P[E]$;

· *Singleton*: $\{\!|-|\!\} : D \to P[D]$, defined by $\{\!|\, d \,|\!\} \stackrel{\text{def}}{=} \{d\}$;

· *Union*: $\uplus : P[D]^2 \to P[D]$, defined by $X \uplus Y \stackrel{\text{def}}{=} Con(X \cup Y)$;

· *Big union*: $\biguplus : P[P[D] \to P[D]]$, defined by: $\biguplus(\Theta) \stackrel{\text{def}}{=} Con(\bigcup \Theta)$;

· *Tensor product*, defined as in [25].

- *Adjoining of the empty set.* The Plotkin powerdomain with empty set, $P^0[D]$, is isomorphic to $(\mathbf{1})_\perp \oplus P[D]$, where $(\mathbf{1})_\perp$ is the lifted one-point domain and $\oplus$ is coalesced sum.

We can now describe the domain equation that determines the domain $\mathcal{D}$ ($\mathcal{D}$ is actually of the form $\mathcal{D}(Act)$, where Act is our carrier set of actions, but, as in [2], we suppress the Act). We define $\mathcal{D}$ to be the initial solution of

$$\mathcal{D} \cong P^0[\sum_{a\in Act} \mathcal{D}].$$

The bottom element of $P^0[\sum_{a\in Act} \mathcal{D}]$ is $\{\!|\perp|\!\}$.

Abramsky provides some considerable detail of the structure of $\mathcal{D}$, most of which we elide. It useful, however, to spell out the finite elements $\mathcal{K}(\mathcal{D}) \subseteq \mathcal{D}$, defined as follows:

- $\emptyset \in \mathcal{D}$
- $\{\!|\perp|\!\} \in \mathcal{K}(\mathcal{D})$
- if $a \in Act$ and $d \in \mathcal{K}(\mathcal{D})$, then $\{\!|\langle a, d\rangle|\!\} \in \mathcal{K}(\mathcal{D})$
- if $d_1, d_2 \in \mathcal{K}(\mathcal{D})$, then $d_1 \uplus d_2 \in \mathcal{K}(\mathcal{D})$.

It is also useful to note that $\mathcal{D}$ can be regarded as a transition system, $\preceq^\mu$, over

Act, as follows:

- $d \xrightarrow{a} d'$ iff $\langle a, d' \rangle \in d$;
- $d \uparrow$ iff $\bot \in d$.

We can then apply our definition of the operational preorder, based on **SCRP**'s operational semantics for each combinator, to $\mathcal{D}$ regarded in this way. We return to consider the structure of this transition system before embarking upon Lemma 5.3.

Taking the ideas summarized so far, we can now describe the first key proposition, 'internal full abstraction' [2]. This result is wholly about the domain $\mathcal{D}$ and does not require any adaptation to handle our resource semantics.

Proposition 5.2 (internal full abstraction) *For all* $d_1, d_2 \in \mathcal{D}$,

$$d_1 \preceq^{\mu} d_2 \text{ iff } d_1 \sqsubseteq d_2.$$

The next step is to interpret **SCRP** in $\mathcal{D}$. Recall that **SCRP**'s judgements are of the form

$$R\,,\, E \xrightarrow{a} R'\,,\, E',$$

where $R' = \mu(a, R)$. **SCRP**'s combinators are interpreted according to the clauses in Table 3, where $\simeq$ denotes Kleene equality, for which we require the following auxiliary definitions:

- The map
$$f : [\mathcal{D}^2 \to \mathcal{D}] \to [(\sum_{a \in Act} \mathcal{D})^2 \to \sum_{a \in Act} \mathcal{D}],$$
used in the clause for product, is defined by
$$f\Phi(x, \bot) = f\Phi(\bot, x) = \bot$$
$$f\Phi(\langle a, d \rangle, \langle b, e \rangle) = \langle ab, \Phi(d, e) \rangle.$$

- The map
$$g_S : [\mathcal{D} \to \mathcal{D}] \to [(\sum_{a \in Act} \mathcal{D}) \to \mathcal{D}],$$
used in the clause for hiding, is defined by
$$g_S\Phi\bot = \{\!|\, \bot \,|\!\}$$
$$g_S\Phi\langle a, d \rangle = \begin{cases} \{\!|\, \langle a, \Phi d \rangle \,|\!\} & \text{if } a \in \mu^{-1}S \\ \emptyset & \text{otherwise.} \end{cases}$$

Note the difference between our definition and the corresponding function in Abramsky's case for restriction: we must unpack the resources S using the inverse image of μ — again, think in terms of 'actions-as-resources'. Note that the clause for the unit action, $\mathbf{1}$, must be given as

$$[\![\mathbf{1}]\!]^{\mathcal{D}}_{\mu}(R) \simeq \lambda f \in \mathcal{D}^{\wp(\mathbf{R})}.\{\!|\, \langle \mathbf{1}, fR \rangle \,|\!\}.$$

$$\begin{aligned}
[\![\Omega]\!]^{\mathcal{D}}_{\mu}(R) &\simeq \{\!|\, \bot \,|\!\} \\
[\![\mathbf{1}]\!]^{\mathcal{D}}_{\mu}(R) &\simeq \lambda f \in \mathcal{D}^{\wp(\mathbf{R})}.\{\!|\, \langle \mathbf{1}, fR \rangle \,|\!\} \\
[\![a : E]\!]^{\mathcal{D}}_{\mu}(R) &\simeq \{\!|\, \langle a\,,\, [\![E]\!]^{\mathcal{D}}_{\mu}\mu(a,R) \rangle \,|\!\} \\
[\![\mathbf{0}]\!]^{\mathcal{D}}_{\mu}R &\simeq \emptyset \\
[\![E + F]\!]^{\mathcal{D}}_{\mu}(R) &\simeq [\![E]\!]^{\mathcal{D}}_{\mu}(R) \uplus [\![F]\!]^{\mathcal{D}}_{\mu}(R) \\
[\![E \times F]\!]^{\mathcal{D}}_{\mu}(R) &\simeq \biguplus_{S \circ T = R}(\mu\Phi \in [\mathcal{D}^2 \to \mathcal{D}]\,.\, f\Phi)([\![E]\!]^{\mathcal{D}}_{\mu}S)([\![F]\!]^{\mathcal{D}}_{\mu}T) \\
[\![(\nu S)E]\!]^{\mathcal{D}}_{\mu}(R) &\simeq (\mu\Phi \in [\mathcal{D} \to \mathcal{D}].\, \biguplus \circ P^0[g_S\Phi])([\![E]\!]^{\mathcal{D}}_{\mu}R \circ S)
\end{aligned}$$

Table 3
The Denotational Semantics of **SCRP**: Clauses for Finite Terms

Note also the clause for concurrent composition: the definition of the interpretation has the form of Day's tensor product [15,16] in $[\wp(\mathbf{R}), \mathcal{D}]$.

Note that our semantics differs from Abramsky's in that it is parametrized on resources. As we have seen, the parametrization is critical for action prefix, concurrent composition, and resource hiding.

Abramsky relates his denotational semantics of the SCCS combinators to the transition relation view of the domain $\mathcal{D}$. For example, with $+^{\mathcal{D}}$ and $\times^{\mathcal{D}}$ denoting the counterparts to $+$ and $\times$ in $\mathcal{D}$,

- $d_1 +^{\mathcal{D}} d_2 \xrightarrow{a} d$ iff $d_1 \xrightarrow{a} d$ or $d_2 \xrightarrow{a} d$, and
- $d_1 \times^{\mathcal{D}} d_2 \xrightarrow{a} d$ iff there exist u_i, v_i, b_i, c_i, for $i = 1, 2$, such that $d_1 \xrightarrow{b_i} u_i$, $d_2 \xrightarrow{c_i} v_i$, for $i = 1, 2$, $(u_1 \times^{\mathcal{D}} v_1) \sqsubseteq d \sqsubseteq (u_2 \times^{\mathcal{D}} v_2)$, and, for $i = 1, 2$, $b_i \# c_i = a$.

We shall also exploit these properties in our setting.

Lemma 5.3 (equivalence of syntactic and semantic terms) *For all finite* **SCRP** *terms R, E, with a given modification map μ, we have R, $E \sim_{\mu} [\![E]\!]^{\mathcal{D}}_{\mu}(R)$.*

We refrain from reconstructing the proof in full detail. We discuss, however, how the argument works in the cases, mentioned above, that vary significantly from Abramsky's treatment of SCCS. The key step is the definition of the measure on process terms used to provide the necessary inductive argument. Abramsky's definition takes a height function ht defined on process terms by

$$\mathsf{ht}(op(E_1, \ldots, E_m)) = 1 + \sup\{\mathsf{ht}(E_i) \mid 1 \leq i \leq m\},$$

where *op* denotes one of the operators of SCCS. In our case, owing to the inter-

dependence between process terms and resource terms, such a measure must be defined on pairs of resources and processes, R, E, in order to give an order, $<$, for the induction. In particular, we need the following cases:

$$\mathsf{ht}(R\,,\, a : E) \;=\; 1 + \mathsf{ht}(\mu(a, R)\,,\, E)$$

$$\mathsf{ht}(R\,,\, E_1 \times E_2) \;=\; 1 + \sup\{\mathsf{ht}(R_1, E_1), \mathsf{ht}(R_2, E_2) \mid R_1 \circ R_2 = R\}$$

$$\mathsf{ht}(R\,,\, (\nu S)E) \;=\; 1 + \mathsf{ht}(R \circ S\,,\, E).$$

Note that we have retained the key property that if $R, E \xrightarrow{a} R', E'$, then $R', E' < R, E$. Thus the induction hypothesis captures the necessary simulations.

We illustrate the argument, which proceeds by induction on the height, ht, and structure of process terms, by considering briefly how the cases for action prefix, product, and hiding can reduced to cases requiring arguments essentially the same as Abramsky's.

- *Action prefix.* In this base case, must show that $R, a : E \sim_\mu [\![a : E]\!]^{\mathcal{D}}_\mu(R)$, but here the parametrization on the resource component evidently has essentially no effect on the underlying (immediate) argument for SCCS: assume, inductively, that $R, E \sim_\mu [\![E]\!]^{\mathcal{D}}_\mu \mu(a, R)$, and the result follows immediately.
- *Product.* We have that if $R\,,\, E \times F \xrightarrow{a} \mu(a, R), E' \times F'$, then, for any b, c such that $a = b\#c$ and S, T such that $R = S \circ T$, for which

  $$S\,,\, E \xrightarrow{b} \mu(b, S)\,,\, E' \text{ and } T\,,\, F \xrightarrow{c} \mu(c, T)\,,\, F'$$

  we have, by the induction hypothesis, that

  $$S\,,\, E \sim_\mu [\![E]\!]^{\mathcal{D}}_\mu S \text{ and } T\,,\, F \sim_\mu [\![F]\!]^{\mathcal{D}}_\mu T.$$

  But, by definition,

  $$[\![E \times F]\!]^{\mathcal{D}}_\mu(R) \simeq \biguplus_{S\circ T=R}(\mu\Phi \in [\mathcal{D}^2 \to \mathcal{D}]\,.\, f\Phi)([\![E]\!]^{\mathcal{D}}_\mu S)([\![F]\!]^{\mathcal{D}}_\mu T),$$

  and, bearing in mind the transition relation view of sums, the result follows.
- *Hiding.* Let $\hat{a}$ be such that $a = \hat{a}/\Pi\mu^{-1}S$. We have that if $R\,,\, (\nu S)E \xrightarrow{a} \mu(a, R)\,,\, (\nu\mu(a, S))E'$, then $R \circ S\,,\, E \xrightarrow{\hat{a}} \mu(\hat{a}, R \circ S)E'$. By the induction hypothesis, we have that

  $$R \circ S\,,\, E \sim_\mu [\![E]\!]^{\mathcal{D}}_\mu(R \circ S),$$

  and the result follows.

Thus, following Abramksy's development, with the essential variations that we described and which are required to accomodate our resource semantics, we obtain the following:

$$[\![X]\!]_\mu^{\mathcal{D},I}(R) \simeq I(X)R$$

$$[\![\mathrm{fix}X.E]\!]_\mu^{\mathcal{D},I}(R) \simeq (\mu d \in [\wp(\mathbf{R}) \to \mathcal{D}] \,.\, [\![E]\!]_\mu^{\mathcal{D},I[d/X]})R$$

Table 4
The Denotational Semantics of **SCRP**: Clauses for Recursive Terms

Theorem 5.4 (full abstraction for finite terms) *For all finite* **SCRP** *terms* R_1, E_1 *and* R_2, E_2*, with a given modification map* μ*,*

$$R_1, E_1 \preceq^\mu R_2, E_2 \quad \textit{iff} \quad [\![E_1]\!]_\mu^{\mathcal{D}}(R_1) \sqsubseteq [\![E_2]\!]_\mu^{\mathcal{D}}(R_2).$$

The extension of this result to recursive terms again follows the pattern of the corresponding result in [2].

We extend our semantics $[\![E]\!]_\mu^{\mathcal{D}} : \wp(\mathbf{R}) \to \mathcal{D}$ to recursively defined terms in the usual way. We introduce a parametrization on an environment $I : Proc \to [\wp(\mathbf{R}) \to \mathcal{D}]$ which leaves the semantic clauses for the finite terms, as given in Table 3, unchanged and which permits the interpretation of recursive terms, as given in Table 4.

Again, following Abramsky's development, with the variations that we have described, we obtain the following:

Theorem 5.5 (full abstraction for recursive terms) *For all* R_1, E_1 *and* R_2, E_2*,*

$$R_1, E_1 \preceq^\mu R_2, E_2 \quad \textit{iff} \quad [\![E_1]\!]_\mu^{\mathcal{D},I}(R_1) \sqsubseteq [\![E_2]\!]_\mu^{\mathcal{D},I}(R).$$

6 A Logic for Resources and Processes

We present briefly the background to, and basic details of, the modal logic **MBI**, introduced in [41].

Kripke models [30,31], based on preordered sets $(W, \sqsubseteq)$, provide a semantic structure rich enough to give meaning to the intuitionistic, classical, and modal connectives. For example, intuitionistically, essential use of the order is made in order to give meaning to implication,

$$v \models \phi \supset \psi \text{ iff, for all } v \sqsubseteq w,\ w \models \phi \text{ implies } w \models \psi.$$

and, classically, the order is used to give meaning to modality,

$$v \models \Box\phi \text{ iff, for all } v \sqsubseteq w,\ w \models \phi.$$

The additional monoidal structure available in Kripke resource monoids allows us to define, in addition to the additive connectives defined as above, a range of

multiplicative connectives. For example,

$$r \models \phi_1 * \phi_2 \text{ iff there are } s_1 \text{ and } s_2 \text{ such that } s_1 \circ s_2 \sqsubseteq r, \text{ and}$$
$$s_1 \models \phi_1 \text{ and } s_2 \models \phi_2.$$

The semantics of the multiplicative conjunction, $*$, is interpreted as follows: the resource r is sufficient to support $\phi_1 * \phi_2$ just in case it can be divided into resources s_1 and s_2 such that s_1 is sufficient to support ϕ_1 and s_2 is sufficient to support ϕ_2. The assertions ϕ_1 and ϕ_2 — think of them as expressing properties of programs — *do not share* resources. In contrast, in the semantics of the additive conjunction, $r \models \phi_1 \wedge \phi_2$ iff $r \models \phi_1$ and $r \models \phi_2$, the assertions ϕ_1 and ϕ_2 may *share* the resource m.

Along with the multiplicative conjunction comes a multiplicative implication, $-\!\!*$, given by

$$r \models \phi -\!\!* \psi \text{ iff for all } s \text{ such that } s \models \phi,$$
$$r \circ s \models \psi.$$

The semantics of the multiplicative implication, $-\!\!*$, may be interpreted as follows: the resource r is sufficient to support $\phi -\!\!* \psi$ just in case for any resource s which is sufficient to support ϕ the combination $r \circ s$ is sufficient to support ψ. We can think of the proposition $\phi -\!\!* \psi$ as (the type of) a function and the proposition ϕ as (the type of) its argument. The resources then describe the cost of applying the function to its argument in order to obtain the result. The function and its argument *do not share* resources.

In the context of processes, recall that Hennessey-Milner logic [48] is a modal logic for describing properties, ϕ, of (CCS) processes, E. The basic formulation of the logic is via the semantic judgement

$$E \models \phi,$$

in which the transition relation given by the operational semantics of the process terms determines the ordering of the worlds. The interaction between the process dynamics and the propositional structure is effected by the semantics of the action modalities. For example,

$$E \models \phi \wedge \psi \text{ iff } E \models \phi \text{ and } E \models \psi$$

$$E \models [a]\phi \text{ iff for all } E \xrightarrow{a} F,\ F \models \phi.$$

In the context of **SCRP**, with an explicit model of resources, we seek a modal logic, called **MBI**, with a basic semantic judgement of the form

$$R\,,\, E \models \phi,$$

which, relative to a given modification function, μ, is intended to be read as 'property ϕ holds of process E relative to resources R'. Here we have that R is a set of resources, with composition and ordering lifted from the underlying Kripke resource monoid,

$$\mathcal{R} = (\mathbf{R}\,,\, \circ\,,\, e\,,\, \sqsubseteq).$$

We obtain a finer analysis of this judgement than is available in Hennessy-Milner logic. Specifically, we obtain, essentially, the following characterization of parallel composition, denoted by $\times$, as in SCCS, where $\sim_\mu$ is the evident notion of bisimulation,

$$\begin{aligned} R\,,\, E \models \phi_1 * \phi_2 \text{ iff there are } R_1 \text{ and } R_2 \text{ such that } R_1 \circ R_2 = R \\ \text{and there are } E_1 \text{ and } E_2 \text{ such that } E_1 \times E_2 \sim_\mu E, \\ \text{such that } R_1\,,\, E_1 \models \phi_1 \text{ and } R_2\,,\, E_2 \models \phi_2. \end{aligned}$$

That is, as a direct consequence of our formulation, we are able to characterize the concurrent structure of the system, together with its resource-constrained synchronization. Finally, by working with **BI**'s multiplicative quantifiers, we are also able to characterize a notion of local resource, with a corresponding logical construct (see [41] and § 8).

The language of **MBI**, introduced in [41], is summarized below. The intended meanings of the less familiar connectives are discussed in the subsequent text; the formal semantics of all of the connectives is given in Table 5. We take *Act* as the domain of predication and quantification; we write $R \circ S\,\downarrow$, *etc.* to emphasise the need for the definedness of the composition. Otherwise, our formulation is based on quite standard methods and so is presented concisely.

- *Atoms*: $p(a_1, \ldots, a_m)$, predication is over actions $a_i \in \mathit{Act}$.
- *Basic Additives*: The classical propositional connectives, $\phi \wedge \psi$, $\top$, $\phi \vee \psi$, and $\bot$.
- *Additive Modalities*: The usual Hennessy-Milner modalities, $[a]\phi$ and $\langle a\rangle\phi$, where $a \in \mathit{Act}$.
- *Additive Quantifiers*: The usual classical quantifiers, $\forall x.\phi$ and $\exists x.\phi$, where the domain of quantification is *Act*.
- *Basic Multiplicatives*: The usual propositional multiplicatives from the bunched logic **BI**, $\phi * \psi$, I, and $\phi \mathrel{-\!\!*} \psi$.
- *Multiplicative Modalities*: Multiplicative forms of the usual Hennessy-Milner modalities, $[a]_\nu\phi$ and $\langle a\rangle_\nu\phi$, where $a \in \mathit{Act}$
- *Multiplicative Quantifiers*: A simple form of the multiplicative quantifiers found in **BI**, $\forall_\nu x.\phi$ and $\exists_\nu x.\phi$, where the domain of quantification is *Act*, in which predication is additive [43,44]. [5]

[5] For binary connectives, such as $*$, one might require that each component of the formula be formed with respect to a different set of variables, with the composite formula requiring the multiplicative combination of the two sets of variables. Here we use the much simpler additive predication, with both components, and so the composite formula, being formed over the same set of variables.

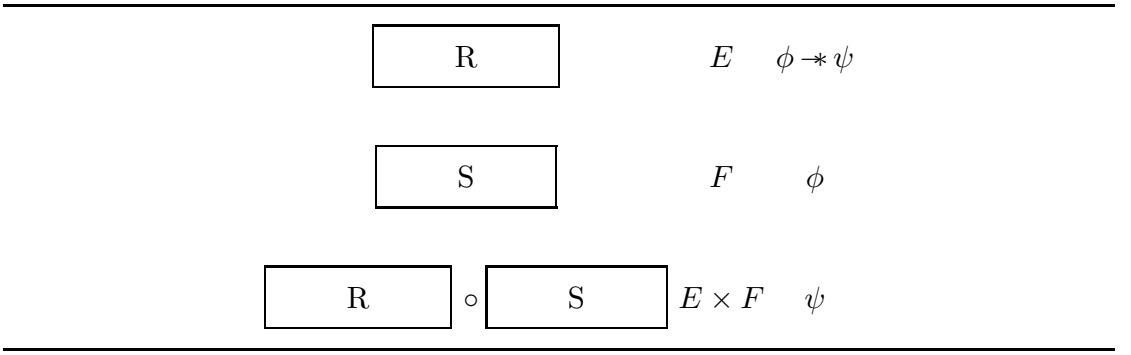

Fig. 2. Multiplicative Resources

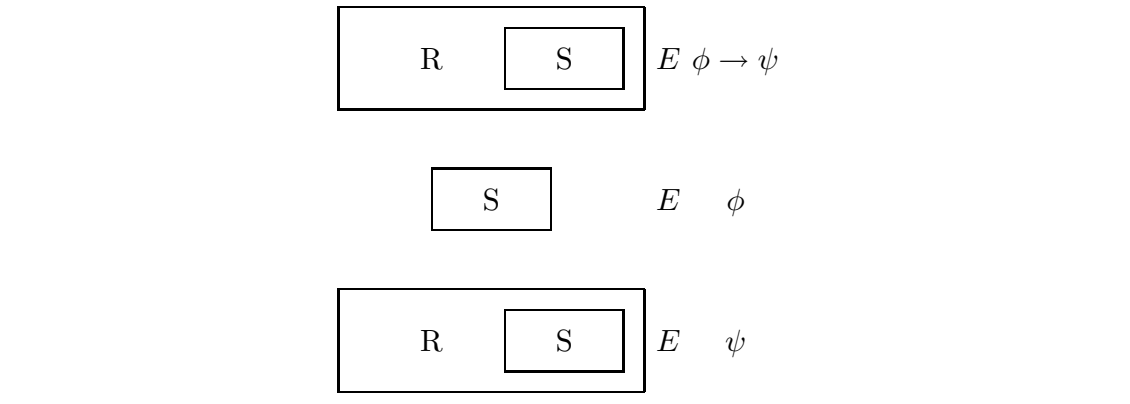

Fig. 3. Additive Resources ($\phi \to \psi \stackrel{\text{def}}{=} (\neg\phi) \vee \psi$)

The clauses for the multiplicative conjunction, $*$, with unit I, and implication, $\mathrel{-\!\!*}$, which establish the basic characterization of concurrent composition described above, are illustrated in Figure 2. Their form follows that taken in the basic formulation of **BI** but we remark that several variations — given by choosing $\sqsubseteq$ or $=$ in each case — are possible. The consequences of these different choices remains to be explored.

The process E which, relative to resource R, has a property given by a multiplicative implication, $\phi \mathrel{-\!\!*} \psi$, imports a module F which, relative to resource S, has a property ϕ. The formula ψ, obtained by eliminating the multiplicative implication, is then a property of the process $E \times F$, relative to the composite resource $R \circ S$. The (classical) additive implication, however, does not give such a characterization of the formation of a concurrent composition. In the usual classical way, it expresses a disjunctive property of (possibly) *shared* resources for a *fixed* process: in Figure 3, S *may* be all of R.

As remarked in [41], it would be possible to take the following Kripke monotonicity, or hereditary, condition:

$$\text{for all } S \text{ s.t. } R \sqsubseteq S,\ R,\ E \models \phi \text{ implies } S,\ E \models \phi.$$

In such a setting, established properties of resource–process pairs would remain true if the available resource were increased. We do *not* take this condition in

general, however. To see why, consider that we might assert that a process has insufficient resource available to evolve even though adding more resource would allow evolution. Such a condition might, however, be derived from the properties of a given monoid. If a monoid, such as the one taken in our illustrative examples, has the property that, for all R, S, $R \sqsubseteq R \circ S$, if the definedness of the modification function is preserved as resource increases, and if the truth of predicate symbols is preserved as resource increases, then, via the clause of Table 5 for atoms, Kripke monotonicity will hold.

As explained in [41], where the basic ideas are introduced, **MBI** makes use not only of the basic multiplicative connectives already discussed but also of multiplicative quantifiers and multiplicative modalities. For example, in a simple resource semantics, we can take

$$R \models \exists_\nu x.\phi \text{ iff for some term } t \text{ defined at } S,$$
$$R \circ S \models \phi[t/x],$$

the point being that new resource is required to form the substituting term. In **MBI**, the additional resource corresponds to that which is hidden by the *Hide* rule, so that the semantic clause for $\exists_\nu$ characterizes hiding up to bisimulation.

The multiplicative modalities are defined similarly. For example, we have that $R, E \models [a]_\nu \phi$ just in case we have that for any action $R \circ S, E \xrightarrow{a} \mu(R \circ S), E'$ — that is, any a that is enabled by additional, separated, resource, S — $R' \circ S', E' \models \phi$. Thus this modality allows us to reason about the additional resource required for actions to occur.

Having discussed the basic ideas of **MBI**, we can now set up the formal definition of models and satisaction.

Let p be an m-ary predicate symbol. Then the interpretation of p in a Kripke resource monoid, $\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq)$,

$$[\![p]\!] : (\wp(\mathbf{R}))^m \to \mathbf{2},$$

is an m-ary relation on $\wp(\mathbf{R})$. With this data, we can give a formal definition of an **MBI** model.

Definition 6.1 [**MBI** model] An **MBI** model is a quadruple

$$\mathcal{M} = \langle \mathcal{R}\,,\, \mu\,,\, [\![-]\!]\,,\, \models^\mu_{\mathcal{M}} \rangle,$$

where $\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq)$ is a Kripke resource monoid, μ is a modification function, $[\![-]\!]$ is an interpretation of the predicate symbols in $\wp(\mathcal{R})$, and $\models^\mu_{\mathcal{M}}$ is a satisfaction relation such the conditions given in Table 5 hold.

Where no confusion can arise, we write just $\models$ rather than $\models^\mu_{\mathcal{M}}$.

The rather simple definition of **MBI** model presented here is adequate for the purposes of this paper. It seems, however, that in order to obtain more delicate

logical results, such as the completeness of a tableaux system (*cf.* [19]), it is necessary to work with more sophisticated classes models, such as those based on ternary relations [18,47,20]. These issues are beyond our present scope.

Notice how, in the clause for atoms in Figure 5, the meaning of an action a corresponds, locally, to the resource for which its modification, $\mu(a, R)$, is defined. Other choices may be possible here.

Before proceeding, in § 8, with the metatheory relating **MBI** and **SCRP**, we return, in the next section, to the basic systems examples described in § 4, to consider them from the perspective of **MBI**.

7 The Systems Examples Revisited

We revisit the examples introduced in § 4 in order to illustrate the interaction between finite resource processes and the polymodal logic **MBI** by checking some simple judgements. Again, we draw directly upon [41]. In a later section, we shall introduce greatest and least fixed points for **MBI** with propositional variables, thereby extending the scope of our logical language to non-finite processes.

The additive connectives correspond to those available in logics of the usual Hennessy-Milner type and are able to express the usual things [34,36]. Accordingly, we concentrate here on examples of the use of the multiplicatives. Once again, we write $\rho(a)$ to denote, where defined, the least resource required to be available for the action a to occur.

Mutual Exclusion

Recall that we define a process in the following manner:

$$E \stackrel{\text{def}}{=} nc : E + critical : E_{critical}$$
$$E_{critical} \stackrel{\text{def}}{=} critical : E_{critical} + critical : E,$$

with $\rho(nc) = \{e\}$ (recall nc is 'not critical'), and $\rho(critical) = \{R\}$. Then the resource process

$$R\,,\, E \times E$$

defines a system exhibiting mutual exclusion.

Recall that, in this example, at no point is the action $critical\#critical$ performed and at no point do we see the state $E_{critical} \times E_{critical}$. Simple (true) assertions about the system include

$$R\,,\, E \times E \models [critical\#critical]\bot.$$

Resource Transfer

As we have seen, the modification functions allow complex notions of resource transfer to be expressed quite naturally. Returning to our simple example of scheduling,

ATOMS

$$R\,,\,E \models_{\mathcal{M}} p(a_1,\ldots,a_m) \text{ iff for all } 1 \le i \le m,\ \mu(a_i,R)\downarrow \text{ and } [\![p]\!](\underbrace{R,\ldots,R}_{m \text{ times}})$$

BASIC ADDITIVES

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \top \qquad \text{always}$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \phi \wedge \psi \text{ iff } R\,,\,E \models^{\mu}_{\mathcal{M}} \phi \text{ and } R\,,\,E \models^{\mu}_{\mathcal{M}} \psi$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \bot \qquad \text{never}$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \phi \vee \psi \text{ iff } R\,,\,E \models^{\mu}_{\mathcal{M}} \phi \text{ or } R\,,\,E \models^{\mu}_{\mathcal{M}} \psi$$

$$R\,,\,E \models_{\mathcal{M}} \neg\phi \text{ iff } R\,,\,E \not\models^{\mu}_{\mathcal{M}} \phi$$

ADDITIVE MODALITIES

$$R\,,\,E \models^{\mu}_{\mathcal{M}} [\,a\,]\phi \text{ iff for all } R, E \xrightarrow{a} \mu(a,R), E' \text{ s.t. } \mu(a,R)\downarrow,\ \mu(a,R)\,,\,E' \models^{\mu}_{\mathcal{M}} \phi$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \langle\,a\,\rangle\phi \text{ iff for some } R, E \xrightarrow{a} \mu(a,R), E' \text{ s.t. } \mu(a,R)\downarrow,\ \mu(a,R)\,,\,E' \models^{\mu}_{\mathcal{M}} \phi$$

ADDITIVE QUANTIFIERS

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \exists x.\phi \text{ iff for some } a \text{ s.t. } \mu(a,R)\downarrow,\ R\,,\,E \models^{\mu}_{\mathcal{M}} \phi[a/x]$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \forall x.\phi \text{ iff for all } a \text{ s.t. } \mu(a,R)\downarrow,\ R\,,\,E \models^{\mu}_{\mathcal{M}} \phi[a/x]$$

BASIC MULTIPLICATIVES

$$R\,,\,E \models^{\mu}_{\mathcal{M}} I \text{ iff } R \sqsubseteq e,\ R\,,\,E \sim_{\mu} R\,,\,\mathbf{1}$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \phi * \psi \text{ iff there exist } S, T \text{ s.t. } S \circ T\downarrow \sqsubseteq R, \text{ and } F, G \text{ s.t. } R\,,\,F \times G \sim_{\mu} R\,,\,E, \text{ and } S\,,\,F \models^{\mu}_{\mathcal{M}} \phi \text{ and } T\,,\,G \models^{\mu}_{\mathcal{M}} \psi$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \phi \mathrel{-\!\!*} \psi \text{ iff for all } S, F \text{ s.t. } R \circ S\downarrow,\ S\,,\,F \models^{\mu}_{\mathcal{M}} \phi,\ R \circ S\,,\,E \times F \models^{\mu}_{\mathcal{M}} \psi$$

MULTIPLICATIVE MODALITIES

$$R\,,\,E \models^{\mu}_{\mathcal{M}} [\,a\,]_{\nu}\phi \text{ iff for all } R \circ S, E \xrightarrow{a} \mu(a, R\circ S), E' \text{ s.t. } R \circ S\downarrow,\ \mu(a,R\circ S)\downarrow, \text{ and } \mu(a,R\circ S)\,,\,E' \models^{\mu}_{\mathcal{M}} \phi$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \langle\,a\,\rangle_{\nu}\phi \text{ iff for some } R \circ S, E \xrightarrow{a} \mu(a, R\circ S), E', \text{ s.t. } R \circ S\downarrow,\ \mu(a,R\circ S)\downarrow, \text{ and } \mu(a,R\circ S)\,,\,E' \models^{\mu}_{\mathcal{M}} \phi$$

MULTIPLICATIVE QUANTIFIERS

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \exists_{\nu} x.\phi \text{ iff for some } S, F \text{ s.t. } R\,,\,E \sim_{\mu} R\,,\,(\nu S)F,\ R \circ S\downarrow,\ R \circ S\,,\,F \models^{\mu}_{\mathcal{M}} \phi[b/x], \text{ for some } b \text{ in } \Pi\mu^{-1}S$$

$$R\,,\,E \models^{\mu}_{\mathcal{M}} \forall_{\nu} x.\phi \text{ iff for all } S, F \text{ s.t. } R\,,\,E \sim_{\mu} R\,,\,(\nu S)F,\ R \circ S\downarrow,\ R \circ S\,,\,F \models^{\mu}_{\mathcal{M}} \phi[b/x], \text{ for all } b \text{ in } \Pi\mu^{-1}S$$

Table 5
Satisfaction for an **MBI** model, $\mathcal{M} = \langle \mathcal{R}\,,\,\mu\,,\,[\![-]\!]\,,\,\models^{\mu}_{\mathcal{M}} \rangle$

again following on from the example above, with process definitions as given in § 4, we have

$$R\,,\, E1 \times E2 \models [get_1 \# get_2]\bot$$

and

$$R\,,\, E1_{critical} \times E2_{critical} \models [put_1 \# put_2]\bot.$$

Defining a rather unsubtle resource-ownership predicate in a system R, E by $owns_{R,E}(a)$ iff an evolution $R, E \xrightarrow{a} \mu(a, R), E'$ occurs, then we get

$$R_1\,,\, E_1 \times E_2 \models [get_1 \# \mathbf{1}]\langle \mathbf{1} \# get_2\rangle owns_{R_2,E_2}(get_2).$$

Handshaking

Recall that the processes

$$E_1 \stackrel{\text{def}}{=} wait_{E_1} : E_1 \,+\, go_{E_1} : E_1'$$
$$E_2 \stackrel{\text{def}}{=} wait_{E_2} : E_2 \,+\, go_{E_2} : E_2'$$

determine a system that can proceed only if they mutually agree on progress: that is, $E_1 \times E_2$ can evolve to $E_1' \times E_2'$ only if $go_{E_1} \# go_{E_2}$ is enabled, that is, R can be decomposed into R_1 and R_2. For $i = 1, 2$, let ϕ_i' be some assertion such that

$$R_i\,,\, E_i' \models \phi_i'.$$

Then we have that

$$R\,,\, E_1 \times E_2 \models \langle go_{E_1} \# go_{E_2}\rangle\,(\phi_1' * \phi_2')$$

provided $R_1 \circ R_2 = R$. This assertion, which demonstrates how a property of a concurrent system may be expressed as a conjunction of properties of its concurrent components, forms part of our next example. Note that if ϕ_2', say, is of the form $\phi_1' \mathrel{-\!\!*} \psi$, then we obtain

$$\langle go_{E_1} \# go_{E_2}\rangle\,\psi$$

as an 'emergent property' of the concurrent system.

Privacy

Recall again that

$$E_1 \stackrel{\text{def}}{=} wait_{E_1} : E_1 \,+\, go_{E_1} : E_1'$$
$$E_2 \stackrel{\text{def}}{=} wait_{E_2} : E_2 \,+\, go_{E_2} : E_2'.$$

For $i = 1, 2$, again let ϕ_i' be such that

$$R_i\,,\, E_i' \models \phi_i'.$$

Then we have that

$$\{e\}\,,\ (\nu R_1 \circ R_2)(E_1 \times E_2) \models \exists_\nu x.\langle x\rangle(\phi_1' * \phi_2'),$$

since unpacking $\models$, using the $\exists_\nu$ clause, gives

$$R_1 \circ R_2\,,\ E_1 \times E_2 \models \langle go_{E_1} \# go_{E_2}\rangle(\phi_1' * \phi_2'),$$

then, using the $\langle -\rangle$ clause,

$$R_1 \circ R_2\,,\ E_1' \times E_2' \models \phi_1' * \phi_2',$$

and finally, using the $*$ clause,

$$R_1\,,\ E_1' \models \phi_1' \quad \text{and} \quad R_2\,,\ E_2' \models \phi_2'.$$

This assertion expresses the property that there exists an action, namely $go_{E_1} \# go_{E_2}$, which is separated from the ambient resources, which allows $E_1 \times E_2$ to evolve locally, using R_1 and R_2 privately, and which leads to a state having the given properties, ϕ_1' and ϕ_2'.

Again, this assertion provides an example of the use of the multiplicative conjunction, $*$, in order to express a property of the concurrent system as a conjunction of properties of its component systems. It also provides an example of the use of the multiplicative existential quantifier, in order to describe a local binding of resources to a component of the system, as well as the more familar diamond modality, $\langle -\rangle$. Notice that the separation condition, between the ambient resource, here $\{e\}$, and the local resource, $R_1 \circ R_2$, is satisfied trivially.

Asynchronous Handover

Recall the producer–consumer system,

$$\begin{aligned} Prod &\stackrel{\text{def}}{=} nowork : Prod + work : Prod \\ Cons &\stackrel{\text{def}}{=} wait : Cons + cons : Cons, \end{aligned}$$

with $\rho(nowork) = \{e\}$, $\rho(wait) = \{e\}$, $\rho(work) = \{e\}$, $\rho(cons) = R$ and, writing R^n for $n > 0$ distinct copies of R, *i.e.*, $\underbrace{R \circ \ldots \circ R}_{n\ times}$,

$$\begin{aligned} \mu(nowork, R^n) &= R^n \\ \mu(wait, R^n) &= R^n \\ \mu(work, R^n) &= R^{n+1} \\ \mu(cons, R^n) &= R^{n-1}. \end{aligned}$$

Let ϕ_{Prod} and ϕ_{Cons} be properties of $Prod$ and $Cons$, respectively, relative to resource R. Then the system $\{e\}, Prod \times Cons$ has the property

$$\{e\}\,,\ Prod \times Cons \models \langle nowork\#cons\rangle_\nu(\phi_{Prod} * \phi_{Cons})$$

since, unpacking $\models$ using $\langle -\rangle_\nu$, noting that

$$\{e\} \circ R, Prod \times Cons \overset{\langle nowork\#cons\rangle}{\rightarrow} \mu(nowork\#cons, \{e\} \circ R), Prod \times Cons$$

gives

$$R\,,\ Prod \times Cons \models \phi_{Prod} * \phi_{Cons},$$

which follows using the case of $\models$ for $*$.

This property says that the system $\{e\}, Prod \times Cons$ may perform the action $nowork\#cons$ provided the required resource be added.

8 Logical Metatheory

Our metatheoretic result relating **SCRP** and **MBI**, taken from [41], is that logical truth in **MBI** models corresponds to bisimulation, extending the logical characterization of bisimulation provided by Hennessy-Milner logic for a process calculus such as CCS [34,36] which takes the form

$$E \sim_\mu F \text{ iff for all } \phi,\ E \models \phi \text{ iff } F \models \phi.$$

In our logically richer setting, we are able to give a finer analysis of the logical characterization of the structure of the process terms. In particular, the connectives of **MBI** can be used to characterize, up to bisimulation, concurrent product and hiding.

To set up this result in our setting, we need to establish some notation.

Definition 8.1 Let Γ be a set of **MBI** formulæ. Then the equivalence $\equiv_\Gamma$ between **SCRP** processes is defined by

$$R\,,\ E \equiv_\Gamma R, F \text{ iff for all } \mathcal{M},\ \{\phi \in \Gamma \mid R\,,\ E \models^\mu_{\mathcal{M}} \phi\} = \{\psi \in \Gamma \mid R\,,\ F \models^\mu_{\mathcal{M}} \psi\}.$$

Following a familiar logical pattern, we have the usual derived definition:

$$R\,,\ E \equiv_{\mathbf{MBI}} R\,,\ F \text{ iff for all } \Gamma,\ R\,,\ E \equiv_\Gamma R\,,\ F.$$

Theorem 8.2 *If, for all R and μ, it is the case that $R\,,\ E \sim_\mu R\,,\ F$, then, for all R, it is the case that $R\,,\ E \equiv_{\mathbf{MBI}} R\,,\ F$.*

Proof. By induction on the structure of formulæ, ϕ. As suggested in § 3, we write R' for $\mu(a, R)$.

p: Let ϕ be $p(a_1, \ldots, a_m)$. By the definition of $\models$, we have that $R, E \models p(a_1, \ldots, a_m)$ iff, for each $1 \leq i \leq m$, $\mu(a_i, R) \downarrow$ and $[\![p]\!](\underbrace{R, \ldots, R}_{m \text{ times}})$. But these conditions are independent of E, so we are done.

$\top$: Similar to the case for atoms, p.

$\wedge$: Let ϕ be $\psi_1 \wedge \psi_2$. By the induction hypothesis, we may assume that the result holds for ψ_1 and ψ_2. By the definition of $\models$, $R, E \models \psi_1 \wedge \psi_2$ iff $R, E \models \psi_1$ and $R, E \models \psi_2$. Therefore, by the induction hypothesis, $R, F \models \psi_1$ and $R, F \models \psi_2$. Therefore, by the definition of $\models$, $R, F \models \psi_1 \wedge \psi_2$.

$\bot$: Similar to the case for atoms, p.

$\vee$: Similar to the case for $\wedge$.

$[a]$: Let ϕ be $[a]\psi$. It follows that, for any E' such that $R, E \xrightarrow{a} R', E'$, $R', E' \models \psi$. By the definition of bisimulation, we have that, for some E' such that $R, E \xrightarrow{a} R', E'$, there is an $R, F \xrightarrow{a} R', F'$ such that $R', E' \sim_\mu R', F'$. So, by the induction hypothesis, $R', F' \models \psi$, and so, by the definition of $\models$, $R, F \models [a]\psi$.

$\langle a \rangle$: Similar to the case for $[a]$.

$\exists$: Let ϕ be $\exists x.\psi$. By the induction hypothesis, we may assume that the result holds for each $\psi[a/x]$, where $a \in Act$. By the definition of $\models$, we have that $R, E \models \exists x.\psi$ iff, for some a such that $\mu(a, R) \downarrow$, $R, E \models \psi[a/x]$. Therefore, by the induction hypothesis, we have that $R, F \models \psi[a/x]$, and the result follows.

$\forall$: Similar to the case for $\exists$.

I: Let ϕ be I. By the definition of $\models$, we have that $R, E \models I$ iff $R \sqsubseteq e$ and $R, E \sim_\mu R, \mathbf{1}$. But if $R, E \sim_\mu R, F$, then $R, F \sim_\mu R, \mathbf{1}$, and the result follows.

$*$: Let ϕ be $\psi_1 * \psi_2$. By the induction hypothesis, we may assume that the result holds for ψ_1 and for ψ_2. By the definition of $\models$, we have that $R, E \models \psi_1 * \psi_2$ iff, for some R_1 and R_2 such that $R_1 \circ R_2 \sqsubseteq R$ and some E_1 and E_2 such that

$$R_1 \circ R_2 \,,\, E_1 \times E_2 \sim_\mu R \,,\, E,$$

$R_1, E_1 \models \psi_1$ and $R_2, E_2 \models \psi_2$. So, by the definition of bisimulation, $R_1 \circ R_2 = R$.

Now suppose that $R, E \sim_\mu R, F$. It follows immediately that $R, F \sim_\mu R_1 \circ R_2, E_1 \times E_2$, and so we are done.

$-\!\!*$: Let ϕ be $\psi_1 -\!\!* \psi_2$. By the induction hypothesis, we may assume that the result holds for ψ_1 and ψ_2. By the definition of $\models$,

$$\begin{aligned} R \,,\, E \models \psi_1 -\!\!* \psi_2 \text{ iff for all } R' \text{ and } E' \text{ such that } R \circ R' \downarrow \\ \text{and } R' \,,\, E' \models \psi_1, \\ R \circ R' \,,\, E \times E' \models \psi_2 \end{aligned}$$

Since $R\,,\ E \sim_\mu R\,,\ F$ and since $\sim_\mu$ is a congruence, we have

$$\begin{aligned} R, E \models \psi_1 \mathrel{-\!\!*} \psi_2 \text{ iff for all } R' \text{ and } E' \text{ such that } R \circ R' \downarrow \\ \text{and } R'\,,\ E' \models \psi_1, \\ R \circ R'\,,\ F \times E' \models \psi_2 \\ \text{iff } R\,,\ F \models \psi_1 \mathrel{-\!\!*} \psi_2 \end{aligned}$$

$[a]_\nu$: Let ϕ be $[a]_\nu\psi$. By the induction hypothesis, we may assume that the result holds for ψ. By the definition of $\models$, we have that $R, E \models [a]_\nu\psi$ iff, for all a and S such that $R \circ S, E \xrightarrow{a} R' \circ S', E'$, subject to some conditions, $R' \circ S', E' \models \psi$. Suppose that $R \circ S, F \xrightarrow{a} R' \circ S', F'$. Then, by the definition of bisimulation, for some E' such that $R \circ S, E \xrightarrow{a} R' \circ S', E'$, $R' \circ S', E' \sim_\mu R' \circ S', F'$. So, by the induction hypothesis, $R' \circ S', F' \models \psi$, and so, by the definition of $\models$, $R, F \models [a]_\nu\psi$.

$\langle a\rangle_\nu$: Similar to the case for $[a]_\nu$.

$\exists_\nu$: Let ϕ be $\exists_\nu x.\psi$. By the induction hypothesis we may assume that the result holds for any $\psi[a/x]$, where $a \in Act$. By the definition of $\models$, we have that $R, E \models \exists_\nu x.\psi$ iff, for some T and some H such that $R, E \sim_\mu R, (\nu T)H$, and some b in $\Pi\mu^{-1}T$,

$$R \circ T\,,\ H \models \psi[b/x],$$

provided $R \circ T \downarrow$. Now suppose that $R \circ T, H' \sim_\mu R \circ T, H$. Then, by the induction hypothesis, we have that

$$R \circ T\,,\ H' \models \psi[b/x] \text{ iff } R \circ T\,,\ H \models \psi[b/x].$$

Now suppose that $R, E \sim_\mu R, F$. It follows immediately that $R, F \sim_\mu R, (\nu T)H$, and so we are done.

$\forall_\nu$: Similar to the case for $\exists_\nu$.

□

We now turn to the converse, that logical equivalence implies bisimulation equivalence. Unfortunately, it does not seem to be possible to use the first-order quantifiers that are naturally present in our system to capture non-image-finite systems. It seems that, just as for Hennessy-Milner logic for CCS, an infinitary version of **MBI**, such as with infinitary additive conjunction, as in [48], is necessary in order to handle the non-image-finite case. Specifically, we conjecture that a version of **MBI** with the evident infinitary $\wedge$, $\mathbf{MBI}_\infty$, will have the property that

$$\text{for all, } R,\ \mu,\ R, E \sim_\mu R, F \text{ iff } R\,,\ E \equiv_{\mathbf{MBI}_\infty} R\,,\ F.$$

For now, however, we establish the basic result for image-finite processes, with the argument following that of Stirling [48] rather straightforwardly. This brings into focus the rôle of the multiplicatives in our setting. They provide a refinement of

the usual analysis of the relationship between logical equivalence and bisimulation equivalence but their absence from the proof of Theorem 8.3 reveals the crudeness of the characterization provided by results of this form.

Theorem 8.3 *If, for all R and μ, R, E and R, F are image-finite and if, for all R, $R\,,\, E \equiv_{\mathbf{MBI}} R\,,\, F$, then, for all R and μ, $R\,,\, E \sim_{\mu} R\,,\, F$.*

Proof. Again, we write R' for $\mu(a, R)$, as suggested in § 3. We adopt Stirling's technique [48] and show that the relation

$$\{(R, E\ ,\ R, F) \mid R, E \text{ and } R, F \text{ image-finite and } R\,,\, E \equiv_{\mathbf{MBI}} R\,,\, F\ \}$$

is a bisimulation.

Seeking a contradiction, we suppose not. Then, without loss of generality, for some R, G and R, H such that $R, G \equiv_{\mathbf{MBI}} R, H$, there are an a and a R', G' such that $R, G \xrightarrow{a} R', G'$ but $R', G' \not\equiv_{\mathbf{MBI}} R', H'$, for all R', H' such that $R, H \xrightarrow{a} R', H'$. Following Stirling's argument, we observe that

$$\mathcal{H} = \{R'\,,\, H' \mid R, H \xrightarrow{a} R'\,,\, H'\}$$

is either empty or not.

Suppose that $\mathcal{H}$ is empty. Then we have both that $R, G \models \langle a\rangle\top$ *and* that $R, H \not\models \langle a\rangle\top$, so contradicting $R, G \equiv_{\mathbf{MBI}} R, H$.

Otherwise $\mathcal{H}$ is non-empty but finite, since we have assumed image-finiteness. So let it be $\{R, H_i \mid 1 \leq i \leq m\}$.

Suppose that, for each $1 \leq i \leq m$, $R, G' \not\equiv_{\mathbf{MBI}} R, H_i$. Then, for each i, there is some $\phi_i(a_i)$, for some $a_i \in Act$, such that $R, G' \models \phi(a_i)$ but $R, H_i \not\models \phi_i(a_i)$.

Now let ϕ be $\phi_1(a_1) \wedge \ldots \wedge \phi_n(a_n)$. Then $R, G' \models \phi$ but $R, H_i \not\models \phi$. Therefore $R, G \models \langle a\rangle\phi$ and $R, H \not\models \langle a\rangle\phi$, and we have a contradiction. Therefore the relation

$$\{\ (R, E\ ,\ R, F) \mid R, E \text{ and } R, F \text{ image-finite and } R, E \equiv_{\mathbf{MBI}} R, F\ \}$$

is a bisimulation. The result follows. □

9 Fixed Points in MBI

Throughout our examples of basic systems constructions in **SCRP**, we have made extensive use of recursive definitions of processes, expressed using the definitional equality, $C \stackrel{\text{def}}{=} E$, with its operational semantics as given by the rule

$$Con \quad \frac{R\,,\, E \xrightarrow{a} \mu(a, R)\,,\, E'}{R\,,\, C \xrightarrow{a} \mu(a, R)\,,\, E'} \quad C \stackrel{\text{def}}{=} E.$$

In order to describe properties of processes defined in this way, we must add, following Larsen [33] a definitional equality to our modal logic. The solutions of recursive equations are expressed as greatest and least fix points.

The addition of fixed-point operators to **MBI** follows the pattern for the modal μ-calculus [48] and is quite straightforward; accordingly, we give only a brief sketch of the definitions and of the extension of the theorem giving the correspondence between bisimulation equivalence and logical equivalence.

The first step is to add propositional variables, denoted X, Y, Z, *etc.*, to the grammar of propositions. The next step is to extend the grammar of propositions with greatest and least fixed points, $\nu X.\phi$ and $\mu X.\phi$, respectively. For the usual technical reasons, sketched below, we must impose a 'covariance' restriction on the occurrences of variables in recursive modal equations. Specifically, in a recursive equation of the form

$$X \stackrel{\text{def}}{=} \phi(X),$$

we require that every free occurrence of X in ϕ lie within the scope of an even number of negations. Letting RP denote a set of transition-closed resource processes, the key point to understand here is our need, just as for the modal μ-calculus, to maintain monotonicity of the operator

$$\mathsf{T}_{\phi,X} : \wp(\mathsf{RP}) \to \wp(\mathsf{RP}),$$

defined, for any $(\mathsf{R},\mathsf{E}) \subseteq \mathsf{RP}$ by

$$\mathsf{T}_{\phi,X}(\mathsf{R},\mathsf{E}) \stackrel{\text{def}}{=} \{\, (R,E) \in \mathsf{RP} \mid R\,,\, E \models_{I[(\mathsf{R},\mathsf{E})/X]} \,\}.$$

The well-known theorem due to Knaster and Tarski gives greatest and least fixed points of functions

$$f : \wp(\mathsf{RP}) \to \wp(\mathsf{RP})$$

that are monotonic with respect to $\subseteq$. The greatest and least of the operator $\mathsf{T}_{\phi,X}$ correspond to the truth-conditional meanings of the greatest and least fixed point formulæ, $\nu X.\phi$ and $\mu X.\phi$ respectively, given in Table 6. Note that the presence of the multiplicatives — not present in, for example, the modal μ-calculus, has no effect on this requirement (*cf.* the fixed point construction given in [45]).

In order to give the semantic definitions of these new formulæ, we need some auxiliary definitions. Firstly, we need environments, I, that assign to each variable X a bisimulation-closed set of **SCRP** terms. Secondly, we need a notation of the set of **SCRP** processes that satisfy a given **MBI** proposition:

$$\| \phi \|_I^{\mathsf{RP}} = \{\, (R\,,\, E) \in \mathsf{RP} \mid R\,,\, E \models_I \phi \,\},$$

where $\models_I$ is the satisfaction relation of Table 5 and RP is a transition-closed set of **SCRP** terms; we write $(R,E) \in \mathsf{RP}$ to denote the **SCRP** process R,E as a member of the set of processes RP. Finally, note that our interpretation of propositional variables as sets of **SCRP** terms is consistent with the case for atoms in Table 5, which merely makes no reference to the process part of the **SCRP** term.

$$R\,,\,E \models_I X \text{ iff } (R\,,\,E) \in I(X)$$

$$R\,,\,E \models_I \nu X.\phi \text{ iff } (R\,,\,E) \in \bigcup\{(\mathsf{R}\,,\,\mathsf{E}) \subseteq \mathsf{RP} \mid (\mathsf{R}\,,\,\mathsf{E}) \subseteq \| \phi \|^{\mathsf{RP}}_{I[(\mathsf{R}\,,\,\mathsf{E})/X]}\}$$

$$R\,,\,E \models_I \mu X.\phi \text{ iff } (R\,,\,E) \in \bigcap\{(\mathsf{R}\,,\,\mathsf{E}) \subseteq \mathsf{RP} \mid \| \phi \|^{\mathsf{RP}}_{I[(\mathsf{R}\,,\,\mathsf{E})/X]} \subseteq (\mathsf{R}\,,\,\mathsf{E})\}$$

Table 6
Satisfaction for fixed points in an **MBI** model

The semantic clauses for the propositional and modal connectives, and for the first-order quantifiers, remain as given in Table 5.

We can now state an extension of Theorems 8.2 and 8.3 to **MBI** with fixed points (recall that we allow ourselves to suppress the modification function μ in the notation for satisfaction).

Theorem 9.1 *For all R, μ, we have that if $R, E \sim_\mu R, F$, then $R, E \equiv_\Gamma R, F$.*

Proof. Both the indirect and direct arguments discussed in [48] can be adapted to our set-up. The indirect approach requires a version of **MBI** with (additive) second-order propositional quantification. The direct approach requires an analysis of bisimulation closed sets for **SCRP**, showing that for any set RP of **SCRP** processes and any second-order sentence (*i.e.*, no free propositional variables) ϕ of **MBI**, the set $\| \phi \|^{\mathsf{RP}}_I$ of processes $(R, E) \in \mathsf{RP}$ such that $R, E \models_I \phi$ is bisimulation closed. We omit the (essentially standard) details. □

For the converse, in the absence of an infinitary additive conjunction in **MBI** with fixed points, the restriction to image-finiteness is required, just as for Theorem 8.3. The same proof goes through, however.

Theorem 9.2 *Let R, E and R, F be image-finite. If $R, E \equiv_\Gamma R, F$, then $R, E \sim_\mu R, F$.*

Just as for basic **MBI** (*i.e.*, without fixed points) we conjecture that the restriction to image-finite resource processes can be removed if an infinitary additive conjunction be added to **MBI** with fixed points.

Concluding, briefly, with an example, we remark that our framework admits the usual treatment of modalities such as 'until', 'always', and 'eventually'. For example, we obtain 'always ϕ' as

$$\Box\phi \stackrel{\text{def}}{=} \mu X\,.\,\phi \vee (\langle - \rangle \top \wedge [-]X),$$

with X not free in ϕ and with $[-]$ denoting 'next', as usual. Similarly, we obtain 'eventually ϕ' as

$$\Diamond\phi \stackrel{\text{def}}{=} \mu X\,.\,\phi \vee \langle - \rangle X,$$

with X not free in ϕ and $\langle - \rangle$ denotes, as usual, the weak 'next' operator.

Thus we can, for example, express a property of our mutual exclusion example, from §§4 and 7, that it is always the case that a system, which displays mutual exclusion, eventually enters the critical region, as the judgement

$$R\,,\, E \times E \models_I \Box\,(\,\Diamond\,\langle \mathit{critical}\rangle\,\top),$$

where $\rho(\mathit{critical}) = \{e\}$ and $\rho(\mathit{got}) = R$.

10 Systems Revisited: Location

The literature on the theory of computation in general, and on concurrent computation in particular, contains a wide range of approaches to the notion of location, and the range of technical complexity required varies greatly. In the work, described above, related to separation logic [46] and bunched polymorphism [14], for example, the starting point is simply a *set* of (names of) locations.

In both the works of Cardelli and Gordon [13], on ambients, and of Jensen and Milner [29], on bigraphs, the notion of location (and, indeed, of resource) is captured within a behavioural framework involving all the complexity of, for example, the π-calculus [37]. Such an approach represents, perhaps, a quest for a grand unified theory of computational structures. Just as with our motivation for separating resources from processes, our ambitions are more prosaic: we seek a conceptually direct, technology for capturing the various features of systems that are relevant to addressing system-scale questions of performance, integrity, and cost (and so of economic viablity).

Whilst admitting that some modelling tasks might require rather more complex notions of location, we begin here by suggesting a basic framework, in the context of our existing analysis of resources and processes and our modelling philosophy, that provides the essential features needed to begin an analysis.

Recall that our resource process judgements are of the form $R\,,\, E \xrightarrow{a} R'\,,\, E'$, for the operational semantics of **SCRP**, and $R\,,\, E \models \phi$, for the logic **MBI**. We enrich these judgements to have the form

$$L\,,\, R\,,\, E \xrightarrow{a} L'\,,\, R'\,,\, E',$$

read as, 'with resources R at starting location L, the process E evolves to E', resulting in resources R' at finishing location L''. Note that we require a connectivity property between L and L', and that the judgement describes just a local evolution. For this conception to be sensible, it seems, following the same modelling philosophy used to derive our assumptions about resources, that we need

- a notion of *sublocation*, $L \preceq M$,
- *substitution* of locations, $M[L'/L]$, of location L' for a sublocation L of M,
- a notion of *connection* between locations, and
- a *product* of locations.

Sublocations arise from, among other things, the need, typically, for a local evolution to describe what happens to the starting location as a result of the evolution. A substitution is required to ensure that we capture an appropriate compositionality property of systems. A product is needed to capture how concurrent actions may draw upon resources from distinct locations. This idea of location captures both the physical and the virtual.

One simple way to realize these requirements is to take locations to be finite, (directed) graphs. Sublocations arise as subgraphs, substitution is given by replacement of a subgraph by a graph of matching arity — that is, matching (directed) arcs — and product is given by a suitable choice of graph product (there are many, including a categorical product and a range of monoidal products). Two sublocations L and M of a location N are connected — taking due account of directedness as necessary — if there is an arc linking a vertex of L to a vertex of M. We believe that the constructs of Cardelli and Gordon [13] and Jensen and Milner [29] can be considered to satisfy these requirements.

Returning to our development from resource-processes to location-resource-processes, it is clear that we must adapt the formulations of the enabling and modification functions. Recalling the basic form of the axiom case of SCRP's operational semantics, we can see that we require, with μ having the evident type,

$$\frac{}{L\,,\,R\,,\,a : E \xrightarrow{a} L'\,,\,R'\,,\,E}$$

with the following definition: $\mu(a, L, R) = (L', R')$.

Note that this framework permits resources to be associated with a location that is either a single vertex or a whole graph, reflecting the choice of degree of abstraction, and providing a highly flexible framework. In the example of the next section, given in Figure 4, we obtain two distinct locations for the jetty resources implicitly — the ordinary jetties and the secure ones.

11 Access Control and Identity

An important application of the systems modelling technology that we have discussed concerns a range of issues in systems and information security.

We begin by discussing, within the context of **SCRP** and **MBI**, some constructs that naturally handle ideas such as *rôles* and *impersonation* in access control, building on ideas discussed by Abadi *et al.* [1,32]. For example, the idea of principal E in rôle F, or 'E quoting F', can be made precise as a form of non-commutative concurrent composition, $E \propto F$, in our setting:

$$\frac{R\,,\,F \xrightarrow{a} R'\,,\,F' \quad S\,,\,E \xrightarrow{a} S'\,,\,E'}{S\,,\,E \propto F \xrightarrow{a} S'\,,\,E' \propto F'}\; R \sqsubseteq S, \quad S\,,\,E \sim_{\mu} S\,,\,F,$$

Interestingly, the non-commuativity arises rather naturally via our explicit representation of resources, not present in [1]. Note that the bisimulation could be relaxed

```
Cons arrival=negexp(10.0); Cons securearrival=negexp(5.0);
Cons leaving=2.0;
Cons dock=2.0; Cons securedock=3.0;
Cons unload=normal(14,3); Cons secureunload=normal(20,5);
Cons tug=3; Cons securetug=4;
Cons jetty=3; Cons securejetty=4;
Cons simdur=1000;

Res(tugs,tug); Res(securetugs,securetug);
Res(jetties, jetty); Res(securejetties, securejetty);

class boat={ Entity(Boat,boat,arrival);
    getR(jetties,1);
     try [getR(tugs,2)] then
      { hold(dock); putR(tugs,2); hold(unload);
        try [getR(tugs,1)] then
         { hold(2.0); putR(tugs,1);
         }
        etry [getR(securetugs,1)] then
         { hold(2.0); putR(securetugs,1);
         }
      }
     etry [getR(securetugs,2)] then
      { hold(dock); putR(securetugs,2); hold(unload);
        getR(securetugs,1); hold(2.0); putR(securetugs,1);
      }
      putR(jetties,1);
} (***boat***)

class secureboat={ Entity(SecureBoat,secureboat,securearrival);
   getR(securejetties,1);
     getR(securetugs,2); hold(securedock);
     putR(securetugs,2); hold(secureunload);
     getR(securetugs,1); hold(2.0);
     putR(securetugs,1);
   putR(securejetties,1);
} (***secureboat***)

Entity(Boat,boat,0.0); Entity(SecureBoat,secureboat,0.0);
hold(simdur);
close;
```

Fig. 4. Demos2k code for docking boats with access control

to simulation, a choice not readily available in Abadi *et al.*'s calculus of principals. Building on this operational construct, we are able to recover the idea of 'principal E *says* ϕ' as a form of modality in **MBI**, $\{E\}\phi$, associated directly with $\propto$:

$$R\,,\, G \models \{E\}\phi \text{ iff for some } F \text{ s.t. } R, G \sim_\mu R, E \propto F, \text{ we have } R\,,\, F \models \phi.$$

That is, E says ϕ holds for G just in case G is of the form E quoting F and F supports ϕ (all relative to resources R). We can enrich this analysis with our notion of location. Abadi *et al.* proceed to analyze a range of derived constructions, involving ideas such as delegation and certificates. These ideas remain to be explored.

More concretely, we can return to Demos2k and give an example of a very simple model of an access control régime in the context of resource semantics. We envisage a variation of the model of boats and docks given in Figure 1 in which we have both boats that require a secure dock and those that may use an insecure dock. The (complete) Demos2k code is given in Figure 4. To enter or leave a secure dock, a secure tug must be used. Secure tugs may be used to enter or leave insecure docks. Thus we have orderings on the resources — tugs and docks. In the Demos2k code, these orderings are implemented implicitly — Demos2k resources do not have an explicit notion of ordering — within the definitions of the the classes for boats and secure boats.

Implicit in this model is a notion of location (the two docks are, implicitly, separated. It is also possible to see that fairly simple modifications of this model could be used to illustrate notions of impersonation from the process-theoretic point of view.

Further work in this area might include the development of tools in the style of Demos2k that embody more aspects of, for example, our accounts of resource and location.

12 Towards a Field Theory

The observation that within **SCRP** we have the property that

$$\text{if } R\,,\, E \xrightarrow{a} R'\,,\, E', \text{ then } R' = \mu(a, R)$$

offers an intriguing possibility for how to capture the behaviour of the system. For some systems — for which we do not yet have a characterization — the complete behaviour could be captured by the modification (*i.e.*, μ) function on resources. We can postulate a simple *functional* calculation of the behaviour of the system:

- Take the intitial resource R;
- For each a such that $\mu(a, R)$ is defined, calculate reachable R's;
- Recurse, possibly infinitely, until the system is closed.

This will provide a *resource* transition system with transitions of the form $R \xrightarrow{a} R'$. It is immediate that this is the largest possible transition system that could be

built from any R, E under μ, and consequently safety properties of the R, E system will be preserved. The establishment of clear links between the structure of E and the consequent filtering effects on the resource-level transition effects may give very efficient approaches to large-scale conditionally presented concurrent systems.

Acknowledgments

We are variously grateful to Samson Abramsky, Graham Birtwistle, Matthew Collinson, Didier Galmiche, Dominique Larchey-Wendling, and Daniel Méry, as well as the anonymous referees and the editors, for their advice and comments on this work.

References

[1] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.

[2] S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92(2):161–218, 1991.

[3] M. Ben-Ari. *Principles of Concurrent and Distributed Programming*. Prentice Hall, 1990.

[4] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In *Proc 11th ICALP, LNCS 172*, 1984.

[5] G. Birtwistle. *Demos — discrete event modelling on Simula*. Macmillan, 1979.

[6] G. Birtwistle. Demos implementation guide and reference manual. Technical Report 81/70/22, University of Calgary, 1981.

[7] G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part I πDemos. *Transactions of the Society for Computer Simulation*, 10(4):299–333, 1993.

[8] G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part II μDemos. *Transactions of the Society for Computer Simulation*, 11(4):303–336, 1994.

[9] G. Birtwistle and C. Tofts. A denotational semantics for a process-based simulation language. *ACM ToMaCS*, 8(3):281 – 305, 1998.

[10] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part I Practice. *Simulation Practice and Theory*, 8(6-7):377–393, 2001.

[11] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part II ... and Theory. *Simulation Practice and Theory*, 8(6-7):395–414, 2001.

[12] Patrice Brémond-Grégoire and Insup Lee. A process algebra of communicating shared resources with dense time and priorities. *Theoretical Computer Science*, 189(1–2):179–219, 1997.

[13] L. Cardelli and A. Gordon. Anytime, anywhere: modal logics for mobile processes. In *Conference Record of the 27th. Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, New York, Boston, Massachusetts, 2000.

[14] M. Collinson, D. Pym, and E. Robinson. On bunched polymorphism. In *Proc. CSL 05*, number 3634 in LNCS, pages 36–50, 2005.

[15] B. J. Day. On closed categories of functors. In S. Mac Lane, editor, *Reports of the Midwest Category Seminar*, volume 137 of *Lecture Notes in Mathematics*, pages 1–38. Springer-Verlag, Berlin-New York, 1970.

[16] B. J. Day. An embedding theorem for closed categories. In A. Dold and B. Eckmann, editors, *Proceedings of the Sydney Category Seminar 1972/73*, volume 420 of *Lecture Notes in Mathematics*, pages 55–65. Springer-Verlag, Berlin, 1973.

[17] Demos2k. http://www.demos2k.org.

[18] J. M. Dunn. Relevant logic and entailment. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, vol. III: Alternatives to Classical Logic*, number 166 in Synthese Library, pages 117–224. D. Reidel, Dordrecht, Holland, 1986.

[19] D. Galmiche, D. Méry, and D. Pym. Resource Tableaux. In *Proc. CSL 2002*, volume 2471 of *LNCS*, pages 183–199, 2002.

[20] D. Galmiche, D. Méry, and D. Pym. The Semantics of **BI** and Resource Tableaux. *Mathematical Structures in Computer Science*, 15:1033–1088, 2005.

[21] P. Gastin and M. Mislove. A simple process algebra based on atomic actions with resources. *Mathematical Structures in Computer Science*, 14:1–55, 2004.

[22] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. The MIT Press, Cambridge, Mass., and London, England, 1992.

[23] Neil J. Gunther. *The Practical Performance Analyst*. Authors Choice Press (an imprint of iUniverse.com, Inc), 2000. Originally published by McGraw Hill.

[24] M. Hennessy. A term model for synchronous processes. *Information and Control*, 51(1):58–75, 1981.

[25] M. Hennessy and G. Plotkin. Full abstraction for a simple parallel programming language. In J. Beçvar, editor, *Mathematical Foundations of Computer Science*, volume 74 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[26] C. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[27] S.S. Ishtiaq and P. O'Hearn. **BI** as an assertion language for mutable data structures. In *28th ACM-SIGPLAN Symposium on Principles of Programming Languages, London*, pages 14–26. Association for Computing Machinery, 2001.

[28] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[29] O.H. Jensen and R. Milner. Bigraphs and mobile processes (revised). Technical report, University of Cambridge, 2004. UCAM-CL-TR-580, ISSN 1476-2986.

[30] S. A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[31] S. A. Kripke. Semantical analysis of intuitionistic logic I. In J. N. Crossley and M. A. E. Dummett, editors, *Formal Systems and Recursive Functions*, pages 92–130. North-Holland, Amsterdam, 1965.

[32] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, 1992.

[33] K. Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72:265–288, 1990.

[34] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer Verlag, 1980.

[35] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.

[36] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[37] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[38] P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.

[39] G. D. Plotkin. A powerdomain construction. *SIAM J. on Computing*, 5:452–487, 1976.

[40] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Dept., Aarhus University, Aarhus, Denmark, 1981.

[41] David Pym and Chris Tofts. A calculus and logic of resources and processes. *Formal Aspects of Computing*, 18(4):495–517, 2006.

[42] D.J. Pym. On bunched predicate logic. In *Proc. LICS'99*, pages 183–192. IEEE Computer Society Press, 1999.

[43] D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002. Errata and Remarks maintained at: http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf.

[44] D.J. Pym. Errata and Remarks for *The Semantics and Proof Theory of the Logic of Bunched Implications* [43]. Maintained at: http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf, 2004.

[45] D.J. Pym, P.W. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of **BI**. *Theoretical Computer Science*, 315(1):257–305, 2004. Erratum: p. 285, l. -12: ", for some $P', Q \equiv P; P'$ " should be "$P \vdash Q$".

[46] J.C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proc. LICS '02*, pages 55–74. IEEE Computer Society Press, 2002.

[47] R. Routley and R.K. Meyer. The semantics of entailment, II-III. *Journal of Philsophical Logic*, 1:53–73 and 192–208, 1972.

[48] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer Verlag, 2001.

[49] Richard Taylor and Chris Tofts. Modelling, myth vs reality, map vs territory. HP Labs Technical Report HPL-2003-246, 2003.

[50] C. Tofts. Efficiently modelling resource in a process algebra. Technical Report HPL-2003-181, HP Laboratories, Bristol, 2003.